НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Веб-додаток для спілкування»

Виконала:
студентка IV курсу, групи ІП-64
Осипенко Анастасія Василівна _____

Керівник:
асистент каф. ОТ
Регіда Павло Геннадійович _____

Консультант з нормоконтролю:
д.т.н., проф. кафедри ОТ
Сімоненко Валерій Павлович _____

Рецензент: _____

_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	3	
2	А4	ІАЛЦ.466400.001 ВП	Відомість дипломного проекту	1	
3	А4	ІАЛЦ.466400.002 ТЗ	Технічне завдання	2	
4	А4	ІАЛЦ.466400.003 ПЗ	Пояснювальна записка	64	
5	А4	ІАЛЦ.466400.004 Д1	Додаток 1. Схема роботи системи	1	
6	А4	ІАЛЦ.466400.005 Д2	Додаток 2. Схема взаємодії програми	1	
7	А4	ІАЛЦ.466400.006 Д3	Додаток 3. Схема програми	1	
8	А4	ІАЛЦ.466400.007 Д4	Додаток 4. Лістинг додатку	36	

				ІАЛЦ.466400.001 ВП		
	ПІБ	Підп.	Дата			
Розробн.	Осипенко А.В.			Відомість дипломного проєкту	Лист	Листів
Керівн.	Регіда П.Г.				1	1
Консульт.					КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІП-64	
Н/контр.	Сімоненко В.П.					
Зав.каф.						

Пояснювальна записка
до дипломного проєкту
на тему: «Веб-додаток для спілкування»

Київ – 2020 року

АНОТАЦІЯ

Надана бакалаврська дипломна робота присвячена веб-додатку для спілкування. Під час виконання роботи було досліджено ринок на предмет існуючих рішень, проведено аналіз їх можливостей і характеристик. Наведено переваги та недоліки можливих рішень.

Розроблене програмне забезпечення надає користувачеві можливість зареєструватися в системі і користуватися її перевагами в подальшому для спілкування з іншими користувачами.

АННОТАЦИЯ

Данная бакалаврская дипломная работа посвящена веб-приложению для общения. В ходе выполнения работы был исследован рынок на предмет существующих решений, проведен анализ их возможностей и характеристик. Приведены достоинства и недостатки возможных решений.

Разработанное программное обеспечение предоставляет пользователю возможность зарегистрироваться в системе и пользоваться ее преимуществами в дальнейшем для общения с другими пользователями.

ABSTRACT

The given thesis is devoted to a Web application for social communication. In the course of the work, the market was investigated for existing solutions, an analysis of their capabilities and characteristics was carried out. The advantages and disadvantages of possible solutions are given.

The developed software gives the user the opportunity to register in the system and enjoy its benefits in the future to communicate with other users.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність –121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____2020 р.

ЗАВДАННЯ
на дипломний проєкт студентці
Осипенко Анастасія Василівна

1. Тема проєкту «*Веб-додаток для спілкування*», керівник проєкту *асистент кафедри ОТ, Регіда Павло Геннадійович*, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студенткою проєкту *20 травня 2020 р*

3. Вихідні дані до проєкту: *технічна документація*

4. Зміст пояснювальної записки: *аналіз предметної області, огляд існуючих рішень, розробка програмного продукту, опис програмної реалізації.*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): *схема роботи системи, схема взаємодії програми, схема програми.*

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	д.т.н., проф. Сімоненко В. П.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Ознайомлення з темою	10.12.2019-15.12.2019	
2	Написання технічного завдання	15.12.2019-15.01.2020	
3	Вивчення теорії	15.01.2020-28.02.2020	
4	Вибір мов програмування та технологій	01.03.2020-08.03.2020	
5	Написання коду	10.03.2020-10.04.2020	
6	Налагодження програми	11.04.2020-01.05.2020	
7	Виправлення помилок	02.05.2020-10.05.2020	
8	Оформлення пояснювальної записки	11.05.2020-04.06.2020	

Студентка Анастасія ОСИПЕНКО

Керівник Павло РЕГІДА

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломного проєкту.

Технічне завдання
до дипломного проекту

на тему: «Веб-додаток для спілкування»

Київ – 2020

ЗМІСТ

					ІАЛЦ.466400.002 ТЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Осипенко А.В.			Веб-додаток для спілкування	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Регіда П.Г.			Технічне завдання		
Реценз.						КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІП-64	
Н. Контр.		Сімоненко В. П.					
Затвердив							

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Веб-додаток для спілкування».

Область застосування: в повсякденному житті для спілкування із знайомими та близькими.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для виконання є завдання на виконання бакалаврського дипломного проекту, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є створення веб-додатку для спілкування.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є науково-технічна література з теорії і практики програмування, бакалаврські роботи інших студентів, публікації в Інтернеті з даних питань.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програми

- Можливість зареєструватися;
- Можливість створювати пости;
- Можливість коментувати пости.

5.2. Вимоги до програмного забезпечення

- Операційна система Windows 7 (або вище) / MacOS High Sierra (або вище) / Linux дистрибутив.

5.3. Вимоги до апаратної частини обчислювальної системи

- Комп’ютер на базі процесора Intel Core i3 або AMD порівнянної потужності;
- Мінімум 2 Гб оперативної пам’яті;
- Вільне місце на диску не менше 500 Мб.

6. ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Ознайомлення з темою	10.12.2019-15.12.2019
Написання технічного завдання	15.12.2019-15.01.2020
Вивчення теорії	15.01.2020-28.02.2020
Вибір мов програмування та технологій	01.03.2020-08.03.2020
Написання коду	10.03.2020-10.04.2020
Налагодження програми	11.04.2020-01.05.2020
Виправлення помилок	02.05.2020-10.05.2020
Оформлення пояснювальної записки	11.05.2020-04.06.2020

Пояснювальна записка
до дипломного проєкту

на тему: «Веб-додаток для спілкування»

Київ – 2020

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

SNS	Social Networking Service – Служба соціальних мереж
HTML	Hypertext Markup Language – мова розмітки гіпертексту
XML	Extensible Markup Language – розширювана мова розмітки гіпертексту
DOM	Document Object Model – об’єктна модель документа
CSS	Cascading Style Sheets – каскадні таблиці стилів; мова, що відповідає за візуальне представлення документів користувачеві
W3C	World Wide Web Consortium – Всесвітній веб-консорціум
JavaEE	Java Enterprise Edition – набір специфікацій, що описує архітектуру серверної платформи для підприємств
API	Application Programming Interface – програмний інтерфейс, що визначає взаємодії між кількома програмними посередниками
JPA	Java Persistence API – специфікація Java EE API, що дозволяє зберігати Java-об’єкти в базі даних в зручному вигляді
ORM	Object-Relational Mapping – об’єктно-реляційне відображення
MVC	Model-View-Controller – схема розділу даних додатку, інтерфейсу користувача та керуючої логіки на три окремих компонента – Модель, Вигляд та Контролер
JDBC	Java Database Connectivity – з’єднання з базами даних на Java
ODBC	Open Database Connectivity, відкрите з’єднання з базами даних
TCP/IP	Transmission Control Protocol/Internet Protocol – протокол керування передачею/міжмережевий протокол
HTTP	Hypertext Transfer Protocol – протокол передачі гіпертексту
JVM	Java Virtual Machine – віртуальна машина Java

JAR	Java ARchive – Java-архів
IDE	Integrated Development Environment – Інтегроване середовище розробки, ІСР, також єдине середовище розробки, ЄСР
JSX	JavaScript XML
REST	REpresentational State Transfer – архітектура, тобто принципи побудови розподілених гіпермедіа систем

ВСТУП

Потенціал комп’ютерних мереж сприяти нововдосконаленим формам комп'ютерної опосередкованої соціальної взаємодії був запропонований доволі рано. Зусилля щодо підтримки соціальних мереж за допомогою комп'ютерно-опосередкованої комунікації були зроблені у багатьох ранніх онлайн-сервісах, включаючи Usenet, ARPANET та LISTSERV [1]. Багато прототипічних особливостей сайтів соціальних мереж також були присутні в таких Інтернет-сервісах, як Prodigy та CompuServe.

Рання соціальна мережа у всесвітньому павутинні розпочалася у формі узагальнених онлайн-спільнот, таких як Theglobe.com (1995), Geocities (1994) та Tripod.com (1995). Багато з цих ранніх спільнот зосереджувались на зближенні людей для взаємодії один з одним за допомогою чатів та заохочували користувачів обмінюватися особистою інформацією та ідеями через особисті веб-сторінки, надаючи прості у користуванні інструменти для публікації та вільний чи недорогий веб-простір. Деякі спільноти – наприклад, Classmates.com – застосували інший підхід, просто змусивши людей посилатися один на одного електронною поштою.

В кінці 90-х профілі користувачів стали центральною особливістю сайтів соціальних мереж, що дозволяють користувачам складати списки «друзів» та шукати інших користувачів з подібними інтересами. Нові методи соціального нетворкінгу були розроблені до кінця 1990-х, і багато сайтів почали розробляти більш вдосконалені функції для користувачів, що дозволили знаходити та керувати «друзями».

Це нове покоління сайтів соціальних мереж почало процвітати з появою SixDegrees у 1997 році, а потім відкритого щоденника "Open Diary" у 1998 році, а в 2003 році першої соціальної мережі Friendster, і незабаром вони стали частиною інтернет-мейнстріму [3].

Friendster набув великої популярності на Тихоокеанських островах. Як свідчить про швидке зростання популярності сайтів у соціальних мережах, до

2005 року повідомлялося, що Myspace отримує більше переглядів сторінок, ніж Google. Facebook, запущений у 2004 році, став найбільшим сайтом соціальних мереж у світі на початку 2009 року. Термін соціальні медіа був введений і незабаром набув широкого поширення.

Служби соціальних мереж на базі веб-сайтів дозволяють з'єднати людей, які поділяють інтереси та діяльність через політичні, економічні та географічні кордони. За допомогою електронної пошти та обміну миттєвими повідомленнями створюються інтернет-спільноти, де заохочують взаємний альтруїзм шляхом співпраці. Все більше людей звертаються до Інтернету та соціальних медіа для задоволення когнітивних, афективних, особистісних інтегративних, соціальних інтегративних та вільних від напруги потреб. Оскільки Інтернет-технології є доповненням для задоволення потреб, це, в свою чергу, впливає на повсякденне життя, включаючи стосунки, школу, церкву, розваги та сім'ю.

Соціальні мережі все частіше стають метою наукових досліджень. Науковці у багатьох сферах почали досліджувати вплив сайтів у соціальних мережах, досліджуючи, як такі сайти можуть грати роль у питаннях ідентичності, конфіденційності, соціального капіталу, молодіжної культури та освіти. За даними дослідження, проведеного в 2015 році, 63% користувачів у США вважають соціальні мережі своїм головним джерелом новин. Дослідження кіберпсихології, проведене австралійськими дослідниками, продемонструвало, що низка позитивних психологічних результатів пов'язана з використанням Facebook. Ці дослідники встановили, що люди можуть отримати відчуття соціальної зв’язності та приналежності до онлайн-середовища.

Зростання використання соціальних мереж відбувається завдяки студентам, які користуються послугами, що працюють в мережі для стажування та працевлаштування. У багатьох навчальних закладах впроваджено онлайн-каталоги випускників, які служать імпровізованими

соціальними мережами, до яких теперішні та колишні учні можуть звернутися за профорієнтацією. Вони також здатні професійно зв’язатися з іншими та налагодити мережу з компаніями. Крім того, було виявлено, що роботодавці використовують соціальні медіа як спосіб дізнатися про особи та поведінку потенційних працівників.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Поняття служби соціальних мереж

Служба соціальних мереж (SNS) – це онлайн-засіб для створення стосунків з іншими людьми, які поділяють інтерес, передумови чи реальні стосунки. Користувачі служб соціальних мереж створюють профіль із особистою інформацією, фотографіями тощо та формують зв’язки з іншими профілями.

Ці користувачі потім використовують створені зв'язки, щоб розвивати відносини за допомогою обміну контентом, електронної пошти, обміну миттєвими повідомленнями, а також коментарів. Послуги соціальних мереж можна також називати "соціальними мережами" або "соціальними медіа" [2].

1.2. Характеристика SNS

Хоча послуги соціальних мереж можуть мати різні форми, вони поділяють декілька характеристик, таких як усі, що використовують Інтернет. Інші подібні характеристики включають:

- Вміст, створений користувачем, наприклад фотографії, відео та публікації, які інформують інших користувачів про діяльність та інтереси публікатора.
- Можливість з’єднувати людей з усього світу, хоча деякі платформи рекомендують людям знати один одного в реальному житті перед тим, як сполучатися в Інтернеті.
- Вони безкоштовні. Їх бізнес-модель базується на широті членства, тому плата за використання була б контрпродуктивною. Однак залишається можливість того, що якщо мережа зросте достатньо великою та корисною, стягнення плати може бути можливим.

- Вони пов'язують людей із загальними сценаріями, такими як відвідування школи, колег по роботі або просто людей зі спільними інтересами.
- Вони можуть допомогти налагоджувати та розвивати стосунки між людьми, які поділяють професію чи бізнес-мережу.
- Вони можуть бути використані для того, щоб допомогти людям під час пошуку інформації, продуктів, послуг чи ресурсів, що мають до них відношення.

1.3. Огляд існуючих рішень

У пункті розглянуті веб-версії трьох найпопулярніших соціальних мереж станом на 2020 р.

1.3.1. Facebook

Facebook – це веб-сайт, запущений 4 лютого 2004 року Марком Цукербергом. Далі наведено перелік структурних функцій програмного забезпечення та технологій, які можна знайти на веб-сайті Facebook та у мобільному додатку, що доступні користувачам цього сервісу.

1.3.1.1. Стрічка новин (News Feed)

Стрічка новин – це первинна система, за допомогою якої користувачі піддаються впливу вмісту, розміщеного в мережі [4]. Скріншот наведено на рисунку. 1.1.

6 вересня 2006 року було оголошено про появу нової функції домашньої сторінки під назвою News Feed. Спочатку, коли користувачі заходили у Facebook, їм було надано налаштовувану версію власного профілю. Новий макет, навпаки, створив альтернативну домашню сторінку, на якій користувачі бачили постійно оновлюваний список діяльності своїх друзів у Facebook.

Стрічка новин виділяє інформацію, що включає зміни профілю, майбутні події та дні народження, серед інших оновлень.

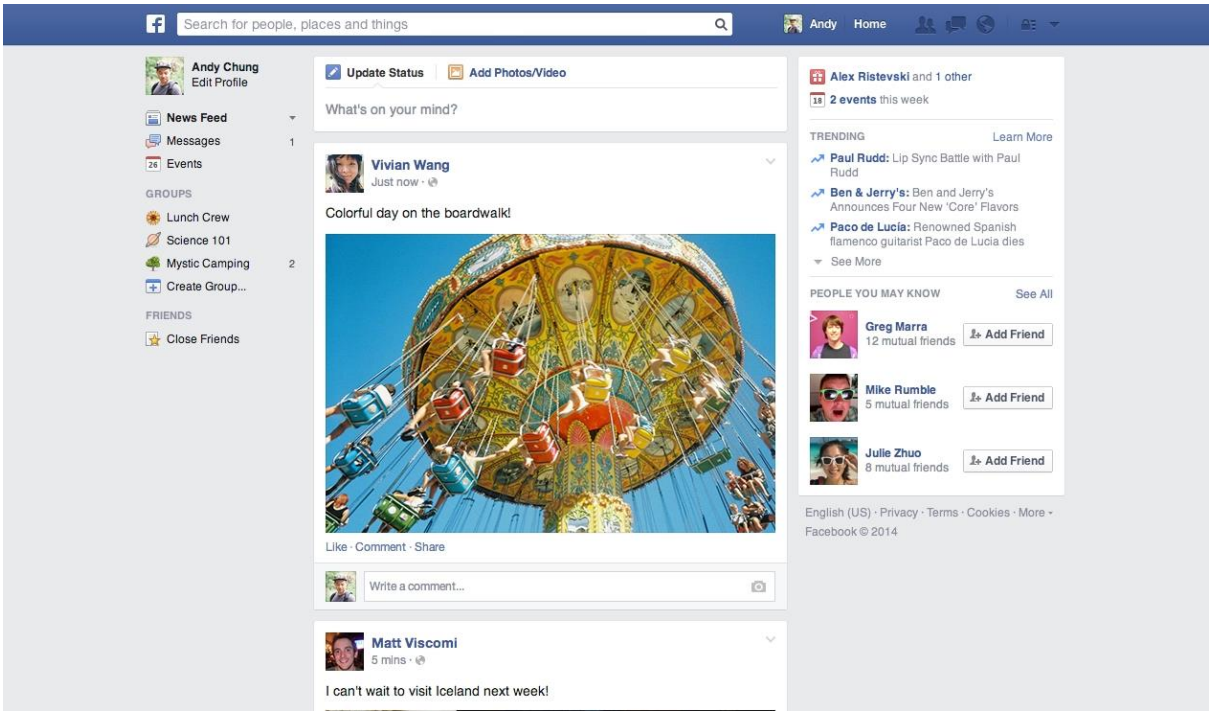


Рис. 1.1. Стрічка новин

Спочатку додавання новостної стрічки викликало певне невдоволення серед користувачів Facebook. Багато користувачів скаржилися, що стрічка занадто захаращена зайвою інформацією. Інші були стурбовані тим, що вона зробила занадто легким для інших людей відслідковувати такі дії, як зміни у стані стосунків, події та розмови з іншими користувачами. У відповідь на це невдоволення творець Марк Цукерберг вибачився за те, що сайт не включив відповідні настроювані функції конфіденційності. Після цього користувачі змогли контролювати, які типи інформації автоматично обмінюються з друзями. Наразі користувачі можуть заважати друзям бачити оновлення щодо декількох видів особливо приватних заходів, хоча інші події не налаштовані таким чином.

1.3.1.2. Друзі

"Френдінг" когось на платформі – це акт надсилання іншому користувачеві "запиту на дружбу" у Facebook. Двоє людей стають друзями у Facebook, як тільки приймаюча сторона приймає прохання про дружбу. Окрім прийняття запиту, користувач має можливість відхилити запит друзів або приховати його за допомогою функції "Не зараз". Видалення запиту про дружбу видаляє запит, але надає відправнику можливість його повторно надіслати в майбутньому. Функція "Не зараз" приховує запит, але не видаляє його, що дозволяє одержувачу переглядати запит пізніше. Скріншот наведено на рисунку. 1.2.

Також на Фейсбуці існує можливість видалити кого-небудь зі списку друзів або заблокувати.

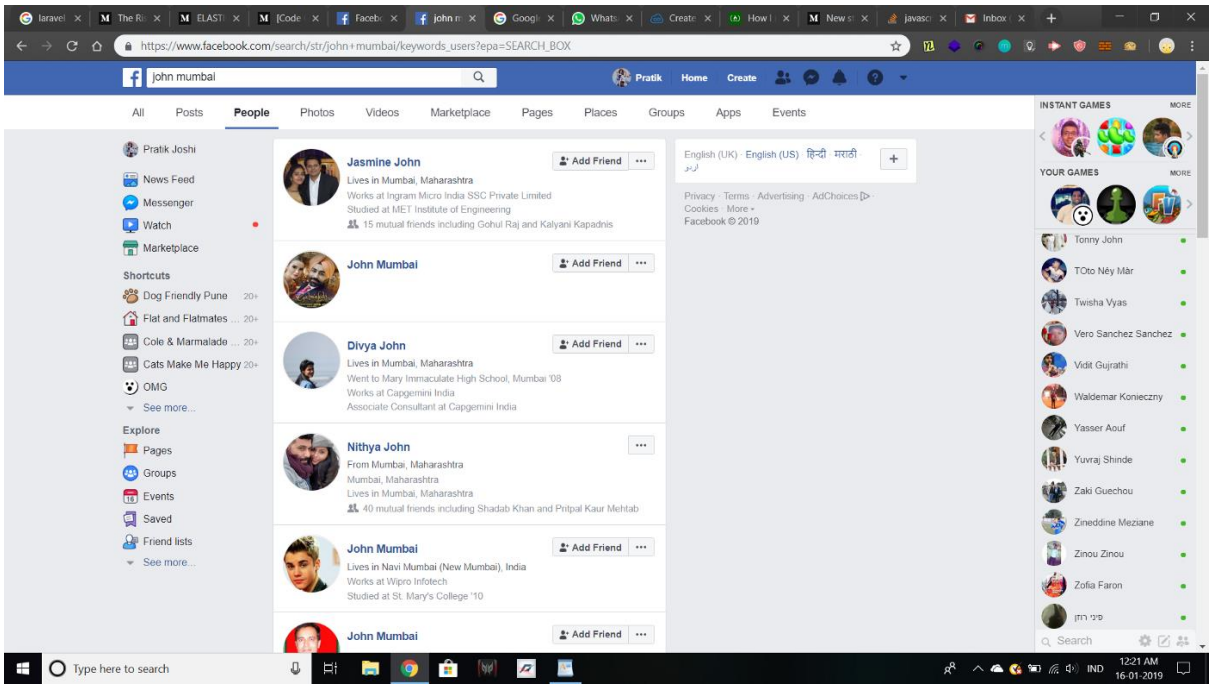


Рис. 1.2. Друзі у Facebook

1.3.1.3. Стіна (Wall)

Стіна – це оригінальний простір профілю, де відображався контент користувачів Facebook до грудня 2011 року. Це дозволяло розміщувати

повідомлення, часто короткі або тимчасові нотатки, на яких користувач міг бачити час та дату написання повідомлення. Стіна користувача була видна всім, хто має можливість бачити його повний профіль, а публікації на стінах друзів відображалися в новинах користувача (дивитись рис. 1.3).

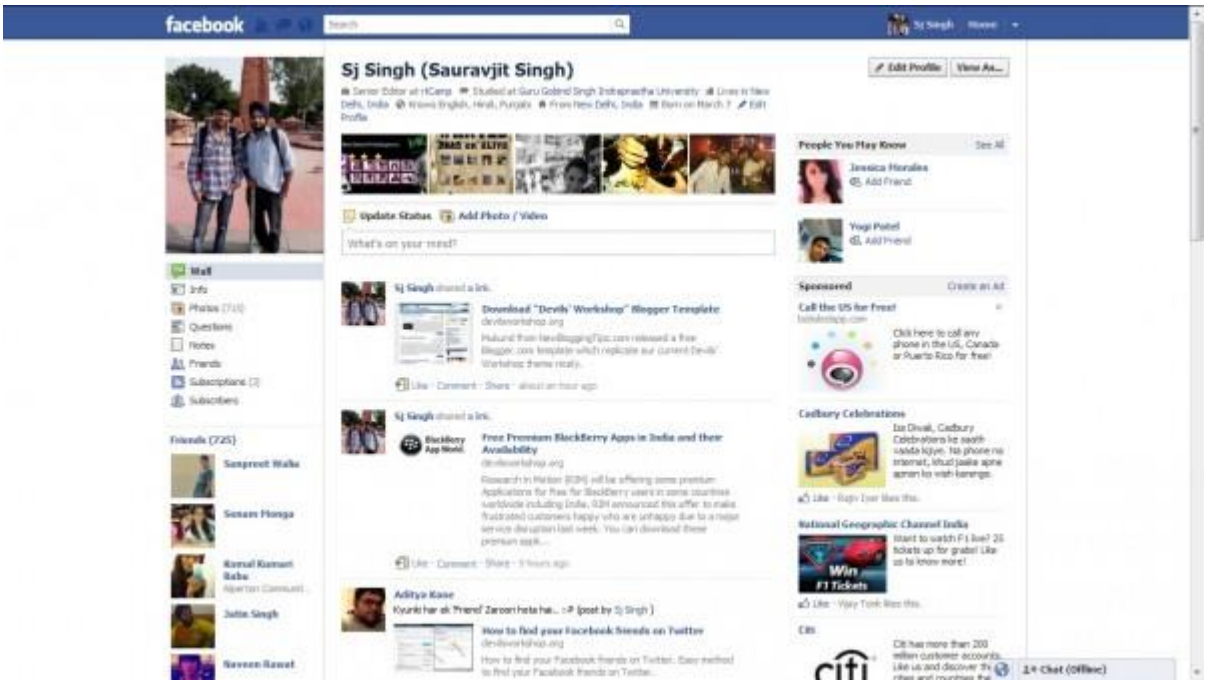


Рис. 1.3. Стіна

Поняття тегів у оновленнях статусу, спроба скопіювати Twitter, розпочалася 14 вересня 2009 р. Це означало ставити ім’я користувача, бренд, подію чи групу у пості таким чином, що цей пост пов'язував теги на стіні сторінки Facebook і робив публікацію видимою у стрічках новин для тегнутої сторінки, а також у публікаціях вибраних друзів. Спочатку це було зроблено за допомогою символу "@", а потім імені людини. Пізніше можна було використати числовий ідентифікатор особи. Візуально це було відображено жирним текстом. На початку 2011 року додано теги в коментарях.

Окрім публікацій інших користувачів, Стіна відображала й інші події, що траплялися з профілем користувача. Це стосувалося ситуацій, коли користувачі змінювали інформацію про себе, своє зображення профілю, а також, коли вони між іншим зв’язувалися з новими людьми.

Стіна була замінена макетом профілю Timeline, який був представлений у грудні 2011 року.

1.3.1.4. Хронологія (Timeline)

У вересні 2011 року Facebook на своїй конференції розробників представив "Timeline", призначену для оновлення профілів користувачів, щоб відображати вміст на основі дати. Скріншот наведено на рисунку. 1.4.

Рис. 1.4. Хронологія

Фотографії з обкладинки були представлені, займаючи значну частину вгорі сторінок, а перероблений показ особистої інформації, наприклад друзів, лайків та фотографій, з’явився зліва, в той час як публікації з історії з'явилися праворуч. Новий дизайн представив гнучкі настроювані розміри для публікацій у стрічці, а також більш помітне розташування та розміщення фотографій. Хронологія також заохочувала прокручування, постійно завантажуючи повідомлення про історії минулих користувачів. Макет сторінки у вигляді хронології почав поступово розповсюджуватися для користувачів у Новій Зеландії, починаючи з 7 грудня 2011 року, і став офіційно

доступним для всіх користувачів у всьому світі 15 грудня. До січня перехід на Timeline став необхідним для всіх користувачів. У лютому 2012 року Timeline стала доступною для сторінок Facebook.

1.3.1.5. Лайки та реакції

Лайк, або кнопка “Подобається” вперше представлена 9 лютого 2009 року, є сигнатурною функцією Facebook, що дозволяє користувачам легко взаємодіяти з оновленнями статусу, коментарями, фотографіями, посиланнями, якими поділяються друзі, відео та рекламою. Після натискання користувачем призначений вміст з'являється у стрічках новин друзів цього користувача, а кнопка також відображає кількість інших користувачів, яким сподобався вміст, включаючи повний або частковий список цих користувачів. Кнопка “Подобається” була поширена на коментарі в червні 2010 року. Після широкого тестування та років запитань громадськості щодо того, чи мав він намір включити кнопку "Не подобається", Facebook 24 лютого 2016 року офіційно розгорнув "Реакції" для користувачів у всьому світі, дозволяючи користувачам довго тримати натиснутою подібну кнопку, щоб скористатися одним із п’яти заздалегідь визначених емоцій, серед яких "Кохання", "Ха-ха", "Нічого собі", "Сумно" або "Злий". Реакції також були поширені на коментарі у травні 2017 року.

1.3.1.6. Коментарі

Щоб відзначити 30-ту річницю GIF (Graphic Interchange Format, картинка, що рухається), Facebook запровадив нову функцію, яка дозволяє користувачам додавати GIF у коментарі. До довгоочікуванної функції можна отримати доступ за допомогою кнопки GIF, розташованої поруч із засобом вибору смайлів. Користувачі можуть вибирати доступні GIF-файли, отримані у партнерів GIF у Facebook, але не можуть завантажувати власні GIF-файли. Скріншот наведено на рисунку. 1.5.

Рис. 1.5. Коментарі

1.3.1.7. Повідомлення

Facebook Messenger – це послуга швидкого обміну повідомленнями та програмне забезпечення. Спочатку розроблена як Facebook Chat у 2008 році, послуга обміну повідомленнями була оновлена у 2010 році, а згодом компанія випустила автономні програми для iOS та Android у серпні 2011 року. Протягом багатьох років Facebook випускала нові додатки для різних операційних систем, запускала спеціалізований інтерфейс веб-сайтів та відокремлювала функціональність обміну повідомленнями від основної програми Facebook, вимагаючи від користувачів завантажувати окремі програми.

1.3.1.8. Сповіщення

Повідомлення надходять користувачеві, коли щось додано на його сторінку профілю. Приклади включають: повідомлення, яким поділились на стіні користувача, коментар до зображення користувача або зображення, яке користувач раніше коментував. Спочатку сповіщення про події обмежувалися

одним на подію; в кінцевому підсумку це були згруповані по категоріях. Наприклад, 10 користувачів, яким сподобалася фотографія користувача, нараховують одне сповіщення, тоді як на більш ранніх етапах вони склали б десять окремих сповіщень. Кількість сповіщень можна змінити в розділі налаштувань до максимуму в 99. У верхній частині сторінки є червоний лічильник сповіщень, який при натисканні відображає останні.

1.3.1.9. Групи

Групи у Facebook можуть створюватися окремими користувачами. Групи дозволяють членам публікувати вміст, такий як посилання, медіафайли, запитання, події, документи та коментарі до цих елементів.

Групи використовуються для колаборації та дозволяють обговорювати події та інші заходи. Вони є способом давати можливість багатьом людям збиратися в Інтернеті для обміну інформацією та обговорення конкретних тем. Групи все частіше використовуються клубами, компаніями та організаціями державного сектору для взаємодії із зацікавленими сторонами, будь то представники громадськості, співробітники, користувачі послуг, акціонери або клієнти. Групи можуть бути публічними або приватними.

Переваги Facebook:

- легка платформа для розпочинання бізнесу
- пропонує зручні інструменти, які допоможуть відстежувати статистику сторінки
- чудова платформа для проведення конкурсів та змагань

Недоліки:

- стає все більш платним
- користувачі являються заручниками системи Лайків та Реакцій
- багато молодих людей віддають перевагу більш новим соціальним мережам

• Facebook найкраще підходить для мобільних пристроїв, що часто може негативно вплинути на сторінку компанії у Facebook

• будучи найбільшою соціальною мережею в світі, все одно має постійні проблеми із приватністю і конфіденціальністю

1.3.2. Twitter

Twitter – це американська соціальна мережа, що базується на мікроблогінгу, комунікація в якій проводиться за допомогою повідомлень, відомих як “твіти”(від англійського “tweet” – щебетати) [5]. Зареєстровані користувачі можуть писати, лайкати, коментувати та репостити твіти, в той час як не зареєстрований користувач може лише переглядати їх. Основні функції Твітера розглянуті далі.

1.3.2.1. Твіти

Твіти публічно видимі за замовчуванням, але відправники можуть обмежувати видимість повідомлень лише для кола своїх підписників. Користувачі можуть твітити через веб-сайт Twitter, сумісні зовнішні програми (наприклад, для смартфонів) або за допомогою SMS, доступною в певних країнах. Користувачі можуть підписатися на твіти інших користувачів – це відоме як "слідування", а підписники відомі як "підписники". Окремі твіти можуть бути переслані іншими користувачами на їх власний профіль, процес відомий як "ретвіт". Користувачі також можуть “вподобати” твіти (раніше “додати в улюблене”). Скріншот наведено на рисунку. 1.6.

1.3.2.2. Механізм хештегів, ретвітів та юзернеймів

Користувачі можуть групувати публікації разом за темою чи типом, використовуючи хештеги – слова чи фрази з префіксом-знаком "#". Аналогічно, знак "@" з іменем користувача використовується для згадування або відповіді інших користувачів. Щоб перезавантажити повідомлення іншого

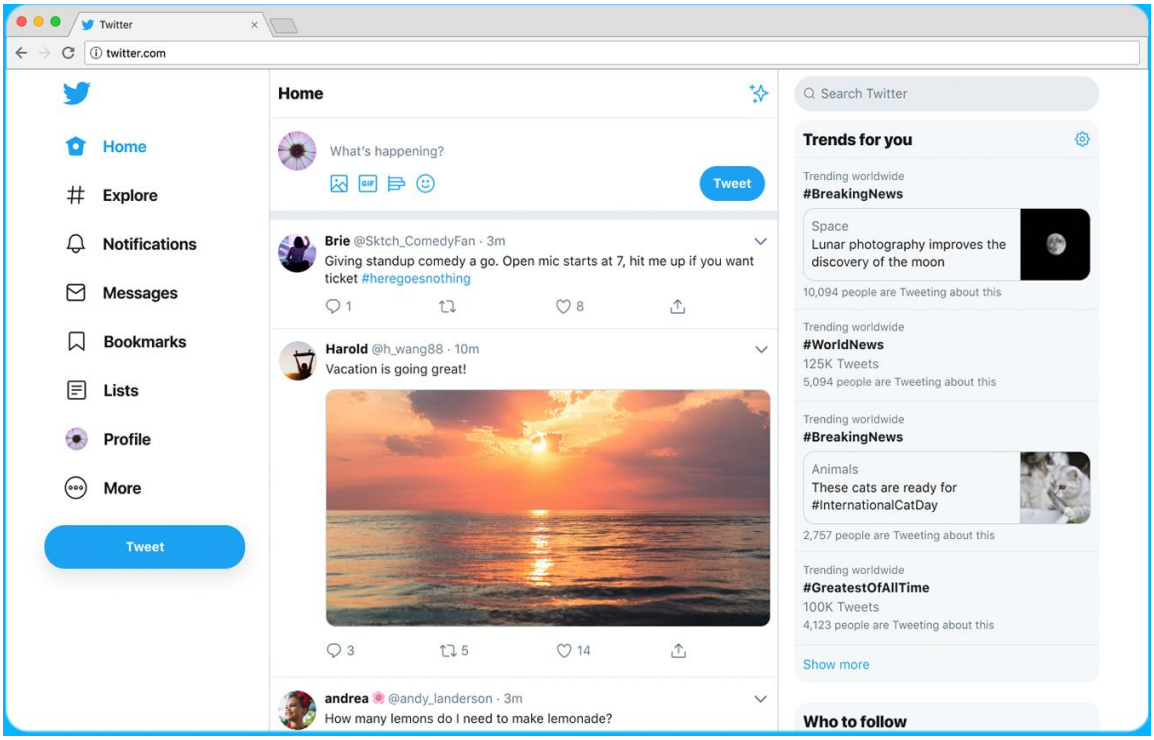


Рис. 1.6. Стрічка Twitter

користувача Twitter і поділитися ним із власними підписниками, користувач може натиснути кнопку ретвіту.

1.3.2.3. Ліміт символів

У 2016 році Twitter оголосив, що такі медіа-елементи, як фотографії та відеозаписи, більше не враховуватимуться у ліміті в 140 символів. Додатки та посилання також більше не будуть частиною обмеження символів.

У 2017 році Twitter подвоїв своє історичне обмеження на 140 символів до 280. Під новим обмеженням гліфи вважаються змінною кількістю символів: більшість європейських літер та пунктуаційних форм вважаються одним символом, тоді як кожний китайський, японський або корейський ієрогліф вважається двома, так що в твіті можна використовувати лише 140 таких символів.

Скріншот наведено на рисунку. 1.7.

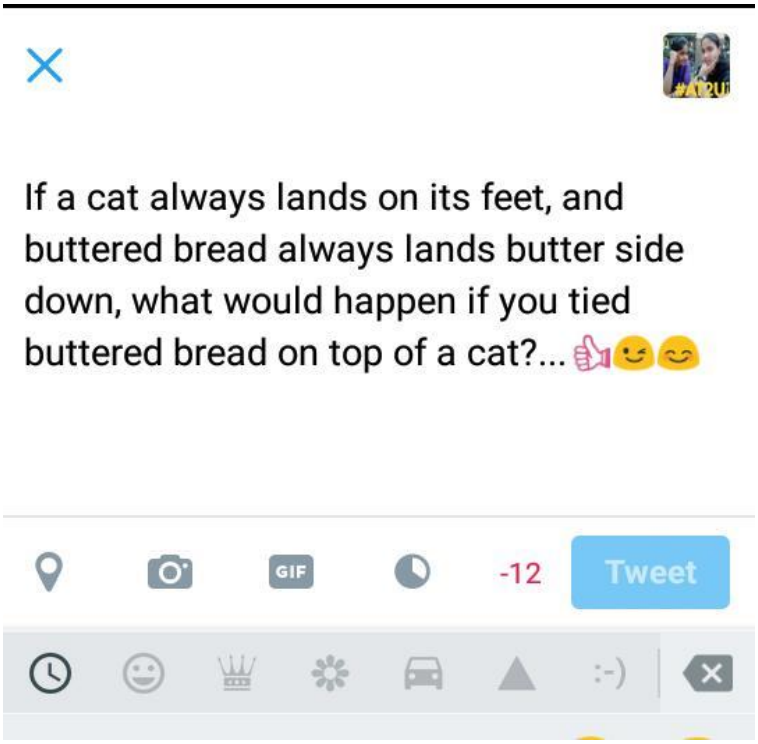


Рис. 1.7. Ліміт символів

1.3.2.4. Тренди

Слово, фразу чи предмет обговорення, про які йдеться більше, ніж про інші, вважають "актуальною темою". Такі теми стають популярними або завдяки узгодженим зусиллям користувачів, або через подію, яка спонукає людей говорити про певну тему. Ці теми допомагають Twitter та його користувачам зрозуміти, що відбувається у світі та якою є думка людей щодо цього.

1.3.2.5. Моменти

У жовтні 2015 року Twitter представив "Моменти" – функцію, яка дозволяє користувачам групувати твіти інших користувачів у більшу колекцію. Спочатку функція була задумана для використання командою розробників; але у вересні 2016 року створення моментів стало доступним для всіх користувачів Twitter [6].

Переваги Twitter:

- Оскільки всі оновлення публікуються в режимі реального часу, можна надсилати повідомлення з більшою частотою
- Забезпечує більше конфіденційності, оскільки підписники можуть твітити до вас так, щоб інші цього не побачили
- Стає все більш ефективною платформою для реклами

Недоліки:

- Маючи обмеження в 140 (280) символів, Twitter не дає дуже багато місця для передавання думки
- Повільний зріст популярності користувачів порівняно з іншими мережами
- Користувачам легко пропустити повідомлення, оскільки все розміщено на хронологічній шкалі

1.3.3. Tumblr

Tumblr (вимовляється "тумблер") – американський веб-сайт, що також базується на мікроблогінгу, заснований у 2007 році. Цей сервіс дозволяє користувачам розміщувати мультимедіа та інший вміст у формі короткого блога. Користувачі можуть переглядати блоги інших користувачів. Блогери також можуть робити свої сторінки приватними. Включає наступні особливості, які розглянуті далі.

1.3.3.1. Панель приладів

Також відома як Dashboard (дивитись рис. 1.8) або інформаційна панель, панель приладів є основним інструментом для типового користувача Tumblr [7]. Це прямий ефір останніх публікацій із блогів, за якими він слідкує. За допомогою інформаційної панелі користувачі можуть коментувати, повторно блокувати і вподобувати публікації з інших блогів, що з’являються

на їх інформаційній панелі. Інформаційна панель дозволяє користувачеві завантажувати текстові повідомлення, зображення, відео, цитати або посилання на свій блог одним натисканням кнопки, що відображається у верхній частині панелі інструментів. Користувачі також можуть підключати

Рис. 1.8. Вкладка Dashboard

свої блоги до своїх акаунтів у Twitter та Facebook; тому щоразу, коли вони публікують публікацію, вона також надсилатиметься як твіт та/або оновлення статусу. Також інформаційна панель дозволяє переглянути кількість власних підписників, постів, постів у черзі та чорновиків (дивитись рис. 1.9) [8].

1.3.3.2. Черга

Користувачі можуть налаштувати графік затримки публікацій, які вони створюють. Вони можуть поширювати свої пости протягом декількох годин або навіть днів.

1.3.3.3. Теги

Користувачі можуть допомогати своїй аудиторії знаходити публікації з певних тем, додаючи теги. Якщо хтось завантажив зображення в свій блог і хотів би, щоб його глядачі знайшли фотографії, вони додали би тег #picture, і

їх глядачі могли б використовувати це слово для пошуку публікацій з тегом #picture.

Рис. 1.9. Частина Dashboard, що відповідає за статистику

1.3.3.4. HTML-редагування

Tumblr дозволяє користувачам редагувати HTML-кодування теми свого блогу, щоб контролювати його зовнішній вигляд. Користувачі також можуть використовувати власне доменне ім’я для свого блогу.

1.3.3.5. Повідомлення

Блогери Tumblr можуть за бажанням дозволити користувачам надсилати запитання, як відкрито, так і анонімно, в блог для відповіді. Tumblr також запропонував функцію "фан-пошти", що дозволяє користувачам надсилати повідомлення в блоги, за якими вони слідкують.

10 листопада 2015 року Tumblr представив інтегровану функцію обміну миттєвими повідомленнями, що дозволяє користувачам спілкуватися з іншими користувачами Tumblr. Платформа обміну повідомленнями замінює систему фан-пошти, яка була застарілою. Можливість надсилати публікації іншим через інформаційну панель була додана наступного місяця.

Переваги Tumblr:

- Приєднавшись до Tumblr, ви отримуєте доступ до вбудованої спільноти, що сприяє зростанню аудиторії легше, ніж на інших веб-майданчиках для блогів
- Tumblr дозволяє планувати публікації, виключаючи потребу в сторонніх програмах

Недоліки:

- Користувачі повинні адаптуватися до формату Tumblr, що є доволі специфічним

Висновки до розділу 1

В розділі 1 було проведено аналіз предметної області, розглянуто поняття служби соціальних мереж (SNS) та її характеристику.

Було проведено огляд існуючих рішень та виокремлено найбільш специфічні особливості, а також переваги та недоліки кожного з рішень.

РОЗДІЛ 2
ПІДГОТОВКА ДО РОЗРОБКИ ПЗ

2.1. Огляд мов програмування для програмної частини додатку

Як зрозуміло з попередніх розділів, програмне забезпечення, яке є кінцевим продуктом даного проєкту – це Веб-застосунок. Тому мова, якою його буде написано, повинна впроваджувати інструменти для веб-розробки, а також бути об’єктно-орієнтованою та кросплатформеною.

Нижче ми наведемо огляд відомих мов програмування:

1. Мова програмування **PHP**. PHP – це вбудована HTML мова скриптів, яка використовується для швидкого формування динамічних веб-сторінок. Чудовий вибір як для розробників фронт-, так і бекенду, щоб додати до свого арсеналу (але особливо для бекенду). Він стоїть за такими веб-гігантами, як WordPress та Facebook. PHP дозволяє швидко та легко розширювати веб-додатки та запускати веб-сайти з повторними серверними завданнями (наприклад, оновлення стрічки). Код PHP є відкритим і користується великою популярністю серед стартап-бізнесу, медіа-агентств та електронної комерції – таких людей, які часто наймають нових розробників веб-сайтів.

2. Мова програмування **Python**. Python є надзвичайно простою для вивчення та динамічною, універсальною мовою. Хоча ця мова більш популярна для бекенду, вона може робити майже все, що ви хочете. Розроблений з метою бути читабельним, простим та найцікавішим, Python – це новий улюбленець розробників у всіх напрямках галузі та є ідеальним для вивчення початківцями. Вона гнучка і надзвичайно потужна і має дуже світле майбутнє.

3. Мова програмування **Javascript**. Мова фронтенду JavaScript, що використовується для створення та розробки веб-сайтів, настільних додатків та ігор. JavaScript працює у всіх браузерах і з ним також можна працювати над програмами, які не базуються на веб-сторінках. Він підтримує як

функціональні, так і об’єктно-орієнтовані стилі програмування, і в основному є заходом для створення інтерфейсів користувача та веб-сайтів/додатків/ігор, які виглядають надзвичайно добре.

4. Мова програмування **Ада.** Мова, що була розроблена військовими з метою автоматизації управління системами реального часу та була серед популярних мов програмування наприкінці XX ст. Проте через наявні конструктивні проблеми, не є найбільш поширеною станом на 2020 р.

5. Мова програмування **C**. C – мова старої школи, легко складена, і має загальне призначення. Це найпоширеніша платформа програмування, яка пропонує побудову елементів для інших мов, таких як C++, Python та Java. Насправді багато з цих мов базуються на C. Чудовий варіант для фулстек розробників і тих, хто хоче додати новий вимір до свого набору навичок. Цю мову найкраще використовувати для написання системного програмного забезпечення та програм, тому це корисна мова для розробників бекенду.

6. Мова програмування **C++**. Загально призначена, добре складена та існуюча приблизно з 1979 року, C++ – це об'єктно-орієнтована, дуже технічна мова. Надзвичайно потужна і з великими бібліотеками, це один із з наріжних каменів розвитку для розробки бекенду. Особливо корисна для програм з високою ефективністю та програм з важкими шаблонами, ця непохитна мова поки що нікуди не дінеться.

7. Мова програмування **Java**. Розроблена в 90-х роках і досі найбільш затребувана мова, Java є золотим стандартом у веб-розробці в усьому світі, у кожній області. Вона є об’єктно-орієнтованою, будується на основі класів і працює на будь-якій платформі, що робить її надзвичайно універсальною. За допомогою Java будуються веб-сайти, мобільні додатки, ігри та інше ПО. Java широко використовується в багатьох галузях промисловості, мало мов розповсюджені так же сильно.

8. Мова програмування **C#.** C# відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java.

Перейнявши багато від своїх попередників – мов C++, Delphi, Modula, Smalltalk і, особливо, Java – C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, C# на відміну від C++ не підтримує множинне успадкування класів (між тим допускається множинне спадкування інтерфейсів).

2.2. Допоміжні фреймворки та бібліотеки

Для прискорення процесу розробки, контролю над зависимостями, а також на заміну Enterprise JavaBeans було обрано Spring Framework, а конкретніше – такі його модулі:

1. **Spring Data JPA.** Налаштування бази даних напряму із коду програми без додаткових схем.

2. **Spring Web MVC.** Дозволяє використовувати Apache Tomcat як дефолтний вбудований контейнер сервлетів.

3. **Spring Security.** Глибоко настроювана аутентифікація та контроль доступу.

4. **Spring Boot DevTools.** Впроваджує більш швидкий запуск та перезапуск проєкту та конфігурації для більш якісного досвіду розробки [9].

2.3. Огляд СУБД

Система управління базами даних, скор СУБД – це сукупність програмних та лінгвістичних засобів загального або спеціалізованого призначення, що забезпечують управління створенням і використанням баз даних. СУБД – комплекс програм, що дозволяють створити базу даних (БД) і маніпулювати даними (додавати, оновлювати, видаляти і вибирати). Система забезпечує надійність зберігання і цілісність даних, а також запроваджує ресурси для адміністрування БД [10].

За моделлю даних серед СУБД вирізняють:

- ієрархічні
- мережеві
- реляційні
- об’єктно-орієнтовані
- об’єктно-реляційні

Для задачі дипломного проєкту найбільш підходить реляційна СУБД. Надалі буде порівняно деякі з них.

2.3.1. MySQL

MySQL – це система управління базами даних, яка дозволяє керувати реляційними базами даних [11]. Це програмне забезпечення з відкритим кодом, що підтримується Oracle. Це означає, що вона є безкоштовною.

Незважаючи на те, що MySQL є програмним забезпеченням з відкритим кодом, можливо придбати комерційну ліцензійну версію від Oracle, щоб отримати послуги преміум-підтримки.

MySQL досить легко освоїти порівняно з іншими СУБД, такими як Oracle Database або Microsoft SQL Server.

MySQL може працювати на різних платформах – MacOS, Linux, Windows тощо. Крім того, MySQL надійна, масштабована та швидка.

2.3.2. PostgreSQL

PostgreSQL – це СУБД загального призначення із об'єктно-реляційною моделлю даних та з відкритим кодом [12]. Його вихідний код доступний під ліцензією PostgreSQL, ліберальною ліцензією з відкритим кодом. Він може бути використаний, змінений та поширений у будь-якій формі.

Спочатку PostgreSQL було розроблено для роботи на UNIX-подібних платформах. Однак пізніше його також було оптимізовано для портативності, щоб він міг працювати на різних платформах, таких як Mac OS X, Solaris та Windows.

Через свою стабільність PostgreSQL вимагає дуже мінімальних зусиль.

2.3.3. IBM Db2

IBM Db2 – це сімейство гібридних продуктів управління даними, що пропонують повний набір можливостей, розширених штучним інтелектом, розроблених, щоб допомогати керувати як структурованими, так і неструктурованими даними на передумовах, а також у приватних та публічних хмарних середовищах. Db2 побудовано на загальному SQL-двигуні, розробленому для масштабованості та гнучкості [13].

Управління даними, що підтримуються штучним інтелектом IBM, зменшує складність, забезпечуючи єдине сімейство Db2, яке працює на базі та створене для штучних інтелектів. Це прискорює їх розвиток, одночасно підвищуючи продуктивність та спритність управління даними.

Завдяки Db2 організації можуть скористатися можливостями штучного інтелекту, введеними в базу даних, для надання прогнозних та активних даних для прийняття рішень. Завдяки Db2 як фундаменту, підприємства мають можливість ефективно використовувати програми, будувати правильну основу для своїх даних та отримувати уявлення про поведінку клієнтів.

2.3.4. H2 Database

H2 – система управління реляційними базами даних, написана на Java. Її можна вбудовувати в програми Java або запускати в режимі клієнт-сервер [14].

Підтримуються наступні режими з'єднання:

● Вбудований режим (локальні з'єднання за допомогою JDBC

● Режим сервера (віддалені з'єднання за допомогою JDBC або ODBC через TCP / IP)

● Змішаний режим (локальні та віддалені з'єднання одночасно)

Вбудований режим (Embedded)

У вбудованому режимі програма відкриває базу даних із JVM за допомогою JDBC. Це найшвидший і найпростіший режим з'єднання. Недоліком є те, що база даних може бути відкрита лише в одній віртуальній машині (і завантажувачі класів) у будь-який час. Як і в усіх режимах, підтримуються як стійкі бази даних, так і ті, що знаходяться в пам’яті. Не існує обмеження кількості одночасно відкритих баз даних або кількості відкритих з'єднань.

Серверний режим

Під час використання серверного режиму (іноді його називають віддаленим режимом або режимом клієнт/сервер), програма відкриває базу даних дистанційно за допомогою API JDBC або ODBC. Сервер потрібно запустити в межах тієї ж чи іншої віртуальної машини або на іншому комп'ютері. Багато додатків можуть підключитися до однієї бази даних одночасно, підключившись до цього сервера. Якщо поглянути внутрішньо, серверний процес відкриває базу даних у вбудованому режимі.

Змішаний режим

Змішаний режим – це поєднання вбудованого та серверного режимів. Перша програма, що підключається до бази даних, робить це у вбудованому режимі, але також запускає сервер, щоб інші програми (що працюють в різних процесах або віртуальних машинах) одночасно отримували доступ до одних і тих же даних. Локальні з'єднання проходять так само швидко, як якщо б база даних використовувалася лише у вбудованому режимі, тоді як віддалені з'єднання дещо повільніші.

2.4. Середа розробки (IDE)

IDE – комплекс програмних засобів, який використовується програмістами для розробки програмного забезпечення [15].

Середовище розробки включає в себе:

- текстовий редактор,

- транслятор (компілятор та/або інтерпретатор),
- засоби автоматизації збирання,
- відладчик.

Іноді ІСР містить також засоби для інтеграції з системами управління версіями і різноманітні інструменти для спрощення конструювання графічного інтерфейсу користувача. Багато сучасні середовища розробки також включають браузер класів, інспектор об'єктів і діаграму ієрархії класів – для використання при об'єктно-орієнтованої розробки ПЗ. ІСР зазвичай призначені для декількох мов програмування – такі як IntelliJ IDEA, NetBeans, Eclipse або Microsoft Visual Studio, але є і IDE для одного певного мови програмування – як, наприклад , Dev-C++.

Використання ІСР для розробки програмного забезпечення є прямою протилежністю способу, в якому використовуються незв'язані інструменти, такі як текстовий редактор, компілятор, і т. п. Інтегровані середовища розробки були створені для того, щоб максимізувати продуктивність програміста завдяки тісно пов'язаним компонентам із простими для користувача інтерфейсами. Це дозволяє розробнику зробити менше дій для перемикання різних режимів, на відміну від дискретних програм розробки. Однак оскільки ІСР є складним програмним комплексом, то середовище розробки зможе якісно прискорити процес розробки ПО лише після спеціального навчання. Для зменшення бар'єру входження багато ІСР є досить інтерактивними, а для полегшення переходу з однієї на іншу інтерфейс у одного виробника максимально близький (приклад – різноманітні середи розробки від JetBrains, до яких відносяться Intellij IDEA та PyCharm).

ІСР зазвичай являє собою єдину програму, в якій проводиться вся розробка. Вона, як правило, містить багато функцій для створення, зміни, компілювання, розгортання і налагодження програмного забезпечення. Мета інтегрованого середовища полягає в тому, щоб об'єднати різні утиліти в одному модулі, який дозволить абстрагуватися від виконання допоміжних

завдань, тим самим дозволяючи програмісту зосередитися на вирішенні власне алгоритмічного завдання і уникнути втрат часу при виконанні типових дій (наприклад, виклику компілятора). Таким чином, підвищується продуктивність праці розробника. Також вважається, що тісна інтеграція завдань розробки може далі підвищити продуктивність за рахунок можливості введення додаткових функцій на проміжних етапах роботи. Наприклад, ICP дозволяє проаналізувати код і тим самим забезпечити миттєвий зворотний зв'язок і повідомити про синтаксичні помилки.

Для розробки було обрано середовище Intellij IDEA Ultimate Edition, тому що ця система має набір інтегрованих інструментів для рефакторінгу, велику базу плагінів, а також програму для студентів, за допомогою якої можливо використовувати всі переваги ICP безкоштовно.

2.5. Огляд технологій для написання клієнтської частини

У пункті розглянуті базові технології, що можуть бути застосовані для ефективної імплементації запропонованого у розділі 2 рішення.

2.5.1. HTML

HTML – це стандартна мова розмітки для документів, призначених для відображення у веб-браузері [16]. Цьому можуть допомогти такі технології, як CSS (Cascading Style Sheets, каскадні таблиці стилів) та мови скриптів, такі як JavaScript.

Веб-браузери отримують HTML-документи з веб-сервера або з локального сховища та переводять документи в мультимедійні веб-сторінки. HTML описує структуру веб-сторінки семантично та спочатку містив підказки для зовнішнього вигляду документа.

Елементи HTML – це складові блоки HTML-сторінок. За допомогою HTML-конструкцій, зображення та інші об'єкти, такі як інтерактивні форми, можуть вбудовуватися у візуалізовану сторінку. HTML забезпечує засіб для

створення структурованих документів, позначаючи структурну семантику для тексту, таких як заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML розмежовані тегами, написаними за допомогою кутових дужок. Такі теги, як <img /> та <input /> безпосередньо вводять вміст на сторінку. Інші теги, такі як <p> оточують і надають інформацію про текст документа, і можуть включати інші теги як під-елементи. Браузери не відображають теги HTML, але використовують їх для інтерпретації вмісту сторінки.

HTML може вбудовувати програми, написані на скриптовій мові, такої як JavaScript, що впливає на поведінку та вміст веб-сторінок. Включення CSS визначає зовнішній вигляд і макет вмісту. Всесвітній веб-консорціум (W3C), колишній утримувач HTML та поточний утримувач стандарту CSS, заохочує використовувати CSS замість презентаційного HTML з 1997 року.

2.5.2. Thymeleaf

Thymeleaf – це сучасний серверний механізм Java на боці сервера як для веб, так і для окремого середовища [17].

Основна мета Thymeleaf – привнести елегантні шаблони у робочий процес розробки – HTML, який може правильно відображатися в браузерах, а також працювати як статичний прототип.

З модулями для Spring Framework, безліччю інтеграцій з різноманітними інструментами та можливістю підключення власного функціоналу Thymeleaf ідеально підходить для сучасної веб-розробки HTML5 JVM в Інтернеті.

2.5.3. Bootstrap

Bootstrap – це вільний набір інструментів для створення сайтів і веб-додатків. Включає в себе HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript- та JQuery-розширення.

2.5.4. Angular

Angular – це фреймворк для дизайну прикладних програм та платформа розробки для створення ефективних та складних односторінкових додатків за допомогою HTML та TypeScript [18]. Цей фреймворк написано мовою TypeScript. Він реалізує основну та додаткову функціональність як набір бібліотек TypeScript, які можна імпортувати у свої програми.

Архітектура програми, написаної за допомогою Angular, спирається на певні фундаментальні концепції. Основними будівельними блоками є NgModules, які забезпечують контекст компіляції компонентів. NgModules збирають відповідний код у функціональні набори; Angular-додаток визначається набором NgModules. У додатку завжди є кореневий модуль, який дозволяє завантажувати основні компоненти, і, як правило, має ще багато функціональних модулів.

Компоненти використовують сервіси, які надають певні функціональні можливості, безпосередньо не пов'язані з переглядами. Постачальники послуг можуть бути введені в компоненти як залежності, що робить код модульним, придатним до багаторазового використання та ефективним.

Як і модулі JavaScript, NgModules можуть імпортувати функціональність з інших NgModules, і дозволяють своїм власним функціям бути експортованими та використаними іншими NgModules. Наприклад, щоб скористатися послугою маршрутизатора у програмі, можна імпортувати NgModule-маршрутизатор.

Організація коду у окремі функціональні модулі допомагає в управлінні розвитком складних додатків та розробці для повторного використання. Крім того, ця методика дозволяє скористатись так званим “ледачим” завантаженням, тобто завантаженням модулів на вимогу, щоб мінімізувати кількість коду, який потрібно завантажити при запуску програми.

2.5.5. React

React (також відомий як React.js або ReactJS) – це бібліотека JavaScript з відкритим кодом для побудови графічних інтерфейсів користувача [19]. Він підтримується Facebook та спільнотою окремих розробників та компаній.

React можна використовувати як базу при розробці односторінкових або мобільних додатків. Однак React стосується лише надання даних у DOM, тому створення React додатків зазвичай вимагає використання додаткових бібліотек для управління станом та маршрутизації.

React робить легшим створення інтерактивних інтерфейсів користувача, дозволяючи створювати прості подання для кожного стану у програмі, до яких React ефективно оновлюватиме та надаватиме лише потрібні компоненти, коли дані зміняться.

Декларативні подання роблять код більш передбачуваним, а налагодження – простішим.

React дозволяє побудувати інкапсульовані компоненти, які керують власним станом, а потім скласти їх для створення складних інтерфейсів.

JSX – це розширення до синтаксису мови JavaScript. За зовнішнім виглядом схожий на HTML, JSX пропонує спосіб структурувати надання компонентів за допомогою синтаксису, знайомого багатьом розробникам. Компоненти React, як правило, записуються за допомогою JSX, хоча вони не повинні цього робити (компоненти також можуть бути записані у чистому JavaScript). JSX схожий на інший синтаксис розширення, створений Facebook для PHP, під назвою XHP.

Ще одна примітна особливість – використання віртуального DOM. React створює кеш-пам'ять структури даних в пам'яті, обчислює отримані відмінності, а потім ефективно оновлює відображений DOM браузера. Цей процес називається примиренням. Це дозволяє програмісту писати код так, ніби вся сторінка відображається при кожній зміні, в той час як бібліотеки React надають лише підкомпоненти, які фактично змінюються. Ця вибіркова

візуалізація забезпечує значне підвищення продуктивності. Це економить зусилля для перерахунку стилів CSS, компонування сторінки та візуалізації для всієї сторінки.

Оскільки логіка компонентів написана на JavaScript замість шаблонів, можна легко передавати дані через додаток і зберігати стани поза межами DOM.

Висновки до розділу 2

В розділі 2 було розглянуто найбільш поширені технології та фреймворки для написання програмного коду та його підтримки.

Проведено огляд систем управління базами даних, оглянуто декілька популярних реляційних СУБД.

Обрано середою розробки Intellij IDEA Ultimate Edition.

Проведено огляд технологій для написання фронтенду для клієнтської частини програми.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1. Загальний огляд проєкту

Проєкт складається з таких частин:

1. Бекенд
2. Веб-інтерфейс користувача
3. Сервер

Як інструмент для контролю над залежностями та збереження певних скриптів, необхідних для збірки і подальшого запуску проєкту, обрано Gradle.

Gradle – це система автоматизації побудови з відкритим кодом, яка ґрунтується на концепціях Apache Ant та Apache Maven і вводить на основі Groovy мову, що орієнтована на домен, замість форми XML, що використовується Maven для оголошення конфігурації проєкту. Gradle використовує спрямований ациклічний граф, щоб визначити порядок виконання завдань.

Gradle був розроблений для багатопроєктних побудов, які можуть стати досить великими. Він підтримує інкрементальні побудови, інтелектуально визначаючи, які частини дерева збірки актуальні; будь-яке завдання, залежне лише від цих частин, не потрібно повторно виконувати.

На рисунку. 3.1 наведено приклад того, що відбудеться після виконання команди gradle build:

```
> gradle build
:compileJava
:processResources
:classes
:jar
:assemble
:compileTestJava
:processTestResources
:testClasses
:test
:check
:build

BUILD SUCCESSFUL
```

Рис. 3.1 Порядок виконання команд при виклику gradle build

compileJava	Компілює вихідні файли Java за допомогою компілятора JDK.
processResources	Копіює виробничі ресурси в каталог виробничих ресурсів.
classes	Це сукупне завдання, яке просто залежить від інших завдань. Інші плагіни можуть додавати до нього додаткові завдання з компіляції.
jar	Збирає файл JAR на основі класів та ресурсів, приєднаних до основного набору джерел.
assemble	Сукупне завдання, яке збирає всі архіви проєкту. Цю задачу додає базовий плагін.
compileTestJava	Компілює тестові вихідні файли Java за допомогою компілятора JDK.
processTestResources	Копіює тестові ресурси в каталог тестових ресурсів.
testClasses	Це сукупне завдання, яке просто залежить від інших завдань. Інші плагіни можуть додавати до нього додаткові завдання з тестування.
test	Виконує unit-тести за допомогою JUnit або TestNG.
check	Сукупне завдання, яке виконує завдання перевірки, такі як виконання тестів. Деякі плагіни додають власні завдання верифікації для перевірки. До цього завдання життєвого циклу слід також приєднати будь-які спеціальні тестові завдання, щоб вони виконувались для повноцінної збірки.
build	Сукупні завдання, які виконують повний збір проєкту. Цю задачу додає базовий плагін.

3.2. Обгрунтування конкретних технологій для розробки

У пункті проведено покроковий аналіз запропонованих у пункті 2.1 технологій та запропоновані практичні переваги та недоліки цих технологій.

3.2.1. Вибір мови програмування

Виходячи із попередніх прикладів у Розділі 2 можна сказати, що для нашої задачі найбільш підходять мови Python, PHP і Java. Інші мови зі списку не зовсім підійшли, тому що вони прив’язані до ОС Windows, у той час як розробка ведеться на ОС Linux. Вибір було зупинено на Java, як одній з найпопулярніших мов для написання будь-яких аплікацій загалом та веб-додатків зокрема. Її використовують майже скрізь і де завгодно. Вона не є залежною від платформи. І найголовніше, вона не дає різного результату в різних системах з різними компіляторами.

Далі буде наведено порівняння Java із попередньо перехованими мовами.

3.2.1.1. Python та Java

Python – мова високого рівня, яка повністю підтримує об'єктно-орієнтоване програмування. Java, з іншого боку, не є чистою об'єктно-орієнтованою мовою.

Python є потужною простою у використанні мовою сценаріїв, яка відрізняється як "клейова" мова, оскільки вона з'єднує системні компоненти, тоді як Java характеризується мовою низького рівня.

Але у Python є деякі недоліки. Програми Python, як правило, запускаються повільніше, ніж програми Java, що робить Java більш привабливим вибором для розробки. Більше того, Java має набагато кращу підтримку бібліотеки для деяких випадків використання, ніж Python.

3.2.1.1. PHP та Java

PHP – це скриптова мова на сервері, тоді як Java – мова загального призначення. Ці дві мови структурно різні.

PHP є слабо типізованою мовою, тоді як Java – це строго типізована мова, де програмісту потрібно оголосити тип даних для кожної змінної та/або

значення. Це може зробити PHP більш привабливим для програмістів, оскільки він не дотримується фіксованих стандартів, як це робить Java, але це, в свою чергу, може ускладнити певні завдання.

Крім структурної різниці, основна різниця між ними полягає в тому, що в PHP віртуальна машина перезапускається після кожного запиту; це може призвести до додаткових проблем із продуктивністю.

PHP належить обирати, якщо немає багато часу для завершення проєкту, але якщо проєкт акцентує увагу на таких функціях, як масштабованість, продуктивність та безпека, слід обрати Java.

Основною мовою програмування було обрано Java версії 1.8, тому що на даний момент вона є однією з найбільш використаних для написання клієнт-серверних додатків, а також через її кросплатформеність.

3.2.2. Вибір СУБД

Для написання бакалаврського проєкту було вирішено використовувати реляційну СУБД H2 Database [20].

H2 – відкрита кроссплатформенна СУБД, написана мовою Java.

Незважаючи на малий розмір (трохи більше 1 МБ) H2 підтримує такі можливості «з коробки»:

- Два режими роботи (клієнт-сервер, вбудований)
- Два режими зберігання даних (файлова система, пам'ять)
- Підтримка планів виконання запитів
- Підтримка кластеризації і реплікації
- Шифрування даних
- Зовнішні (пов'язані) таблиці
- Драйвер ODBC
- Повнотекстовий пошук
- Визначення доменів
- Мультіверсійний конкурентний доступ

- Підтримка послідовностей
- Підтримка ключових слів в запитах
- Тимчасові таблиці
- Обчислювані стовпці
- Призначені для користувача агрегатні функції
- Призначені для користувача процедури, що зберігаються
- Робота з CSV файлами на читання і запис
- Браузерна консоль управління

Нижче рисунку 3.2 порівняльну таблицю бази даних H2 та деяких популярних реляційних систем управління базами даних:

	H2	Derby	HSQLDB	MySQL	PostgreSQL
Pure Java	Yes	Yes	Yes	No	No
Memory Mode	Yes	Yes	Yes	No	No
Encrypted Database	Yes	Yes	Yes	No	No
ODBC Driver	Yes	No	No	Yes	Yes
Fulltext Search	Yes	No	No	Yes	Yes
Multi Version Concurrency	Yes	No	Yes	Yes	Yes
Footprint (embedded)	~2 MB	~3 MB	~1.5 MB	—	—
Footprint (client)	~500 KB	~600 KB	~1.5 MB	~1 MB	~700 KB

Рис. 3.2. Порівняння баз даних

3.2.3. Вибір супровідних бібліотек

Під час написання коду програми знадобляться наступні залежності.

3.2.3.1. Rest Repositories

Spring Data REST є частиною проєкту Spring Data, що полегшує побудову гіпермедіативних веб-сервісів, використовуючи репозиторії Spring Data.

REST репозиторій аналізує доменну модель програми, та впроваджує гіпермедіативні HTTP ресурси для агрегатів, що містяться в моделі.

REST підхід надає такі переваги:

a) Масштабованість взаємодії компонентів системи (додатків)

b) Спільність інтерфейсів

c) Незалежне упровадження компонентів

d) Проміжні компоненти, що можливо знижують затримку та посилюють безпечність

3.2.3.1. JPA

Як специфікація, JPA відповідає за постійність, що означає будь-який механізм, за допомогою якого об'єкти Java переживають процес застосунку, що їх створив. Не всі об’єкти Java потрібно зберігати, але більшість програм зберігають ключові бізнес-об’єкти. Специфікація JPA дозволяє визначити, які об’єкти слід зберігати та як ці об’єкти слід зберігати у додатках Java.

JPA сама по собі не є інструментом або фреймворком; скоріше, він визначає набір концепцій, які можуть бути реалізовані будь-яким інструментом або рамкою. Модель об'єктно-реляційного відображення (ORM) JPA базується на Hibernate.

Через свою переплетену історію JPA та Hibernate доволі часто плутають. Однак, як і Java Servlet, JPA породила багато сумісних инструментів та фреймворків, і Hibernate – лише один з них.

Hibernate - це ORM-бібліотека для Java, яка розроблена як альтернатива entity beans для persistence. На сьогоднішній день Hibernate ORM - одна з найбільш зрілих реалізацій JPA та все ще популярний варіант ORM на Java.

Хоча вони відрізняються між собою у виконанні, кожна реалізація JPA надає певний рівень ORM. Для того, щоб зрозуміти JPA і інструменти, сумісні з JPA, потрібно добре зрозуміти ORM.

Об'єктно-реляційне відображення - це завдання, яке є вагомі підстави уникати робити вручну. Фреймворк на зразок Hibernate ORM кодифікує це завдання в бібліотеку або структуру рівня ORM. Як частина архітектури додатків, рівень ORM відповідає за управління перетворенням об’єктів програмного забезпечення для взаємодії з таблицями та стовпцями у реляційній базі даних. У Java рівень ORM перетворює класи та об'єкти Java, щоб вони могли зберігатися та керуватися у реляційній базі даних.

За замовчуванням ім'я об'єкта, що зберігається, стає назвою таблиці, а поля стають стовпцями. Після встановлення таблиці кожен рядок таблиці відповідає об'єкту в додатку. Відображення об'єктів можна налаштовувати, але типово вони за умовчанням налаштовані добре.

3.3. Вибір технологій для реалізації фронтенд-частини веб-інтерфейсу користувача

Для написання веб-інтерфейсу було обрано наступну комбінацію із розглянутих у попередніх пунктах технологій.

3.3.1. Thymeleaf

1) Двигун шаблонів Java для XML, XHTML та HTML5 [21].

2) Працює як у веб-, так і офлайн-середовищах. Немає жорсткої залежності від сервлетів.

3) Базується на основі модульних наборів ознак – діалектів:

a) Діалектичні функції (наприклад: оцінка, ітерація тощо) застосовуються шляхом посилання їх на теги та / або атрибути шаблону.

b) Наявні два діалекти: Стандартний та стандарт для Spring (для додатків MVC Spring, синтаксис однаковий).

c) Розробники можуть розширювати та створювати власні діалекти.

4) Кілька режимів шаблону:

a) XML: перевіряємий за визначеним типом документа чи без.

b) XHTML 1.0 і 1.1: перевіряємий за визначеним типом документа.

c) HTML5: код, сформований у XML, і заснований на застарілому HTML5. Застарілий не-XML-код буде автоматично підчищений та перетворений у форму XML.

5) Повна, відкрита для розширення підтримка інтернаціоналізації.

6) Налаштований кешований синтаксичний кешований шаблон, що знижує використання вводу/виводу до мінімуму.

7) Автоматичні переклади DOCTYPE для перевірки як шаблону, так і коду результату.

3.3.2. Bootstrap

Як було описано в розділі 2, Bootstrap використовує сучасні напрацювання в області CSS і HTML, тому необхідно бути уважним при підтримці старих браузерів.

Основні інструменти Bootstrap:

Сітки	заздалегідь задані розміри колонок, які можна відразу ж використовувати в CSS-описі документа.
Шаблони	фіксований або “гумовий” шаблон документа.
Типографіка	опис шрифтів, визначення деяких класів для шрифтів, таких як код, цитати і т. п.
Медіа	надає деяке упорядкування зображень та відео.
Таблиці	ресурси для оформлення таблиць, аж до додавання функціональності сортування.
Форми	класи для оформлення форм і деяких подій, що відбуваються з ними.
Навігація	класи оформлення для панелей, вкладок, переходу по сторінках, меню і панелі інструментів.
Алерт	оформлення діалогових вікон, підказок і спливаючих вікон.

3.3.3. React

Код React-додатку складається з об'єктів, що називаються компонентами. Компоненти можуть бути передані певному елементу в DOM за допомогою бібліотеки React DOM. Під час візуалізації компонента можна передати значення, відомі як реквізити.

Два основних способи декларування компонентів у React – це через функціональні компоненти та компоненти на основі класу:

Функціональні компоненти оголошуються функцією, яка потім повертає деякі JSX.

Компоненти на основі класу оголошуються за допомогою класів ES6. Вони також відомі як "стаціонарні" компоненти, тому що їх стан може містити значення у всьому компоненті і може бути переданий дочірнім компонентам через реквізит.

Базова архітектура React застосовується за межами візуалізації HTML у браузері. Наприклад, Facebook має динамічні діаграми, які відображають теги <canvas>, а Netflix і PayPal використовують універсальне завантаження для надання ідентичного HTML на сервері та клієнті.

3.3.4. Webpack

Webpack - це постачальник модулів JavaScript з відкритим кодом. Хоча і створений в основному для JavaScript, він може трансформувати активні елементи, такі як HTML, CSS та зображення, якщо включені відповідні завантажувачі. Webpack приймає модулі із залежностями та генерує статичні активи, що представляють ці модулі.

Webpack приймає залежності та генерує граф залежності, що дозволяє використовувати модульний підхід для розробки веб-додатків. Він може бути використаний з командного рядка, або може бути налаштований за допомогою конфігураційного файлу, який називається webpack.config.js. Цей файл

використовується для визначення правил, плагінів тощо для проєкту. Node.js необхідний для використання Webpack.

3.4. Вбудоване розгортання веб-сервера

Spring Boot використовує загальнодоступну основну точку входу у програму, яка запускає вбудований веб-сервер.

3.4.1. Apache Tomcat

Під час запуску програми Spring Boot виявить, що у вас є контролер Spring MVC, і запустить за замовчуванням вбудований екземпляр Apache Tomcat 7. Протестувати кінцеву точку REST можна, відкривши веб-переглядач та перейшовши за адресою http://localhost:8080/.

Існує безліч варіантів конфігурації вбудованого Tomcat. Досить легко ввімкнути HTTPS (переривання SSL/TLS) для будь-якої користувацької веб-служби, надавши EmbeddedServletContainerCustomizer.

Цей інтерфейс конфігурації дозволяє використовувати більшу частину потужності явної конфігурації XML для окремого екземпляра Apache Tomcat. Дрібні речі, наприклад, на якому порту працює сервер, можна налаштувати, вказавши властивості або через командний рядок, або через завантажений файл властивостей. Таким чином, щоб змінити порт, на якому працює Tomcat, можна відредагувати файл властивостей відповідним чином.

За замовчуванням Spring Boot використовує Tomcat 7. Для використання Tomcat 8 потрібно лише замінити властивість tomcat.version збірки, і це спровокує вибір пізніших збірок Apache Tomcat.

Висновки до розділу 3

У розділі 3 було визначено конкретні технології, мови та бібліотеки, що будуть використовуватись для програмної реалізації додатку.

Визначено інструменти для розробки як клієнтської, так і серверної частин системи; зокрема під час написання коду програми, а також систему управління базами даних.

З метою ефективного оперування інформацією у базах даних, було обрано REST Repositories та JPA, у якості СУБД обрано H2 Database. У якості серверу для деплойменту запропоновано використання Embedded Apache Tomcat.

Для написання клієнтської частини обрано комбінацію із Thymeleaf, Bootstrap, React та Webpack.

РОЗДІЛ 4
МОДЕЛЮВАННЯ ЗАПРОПОНОВАНОГО АЛГОРИТМУ

4.1. Архітектура додатку

Реалізований додаток, так як він є веб-додатком, має архітектуру Клієнт-сервер (дивитись рис. 4.1).

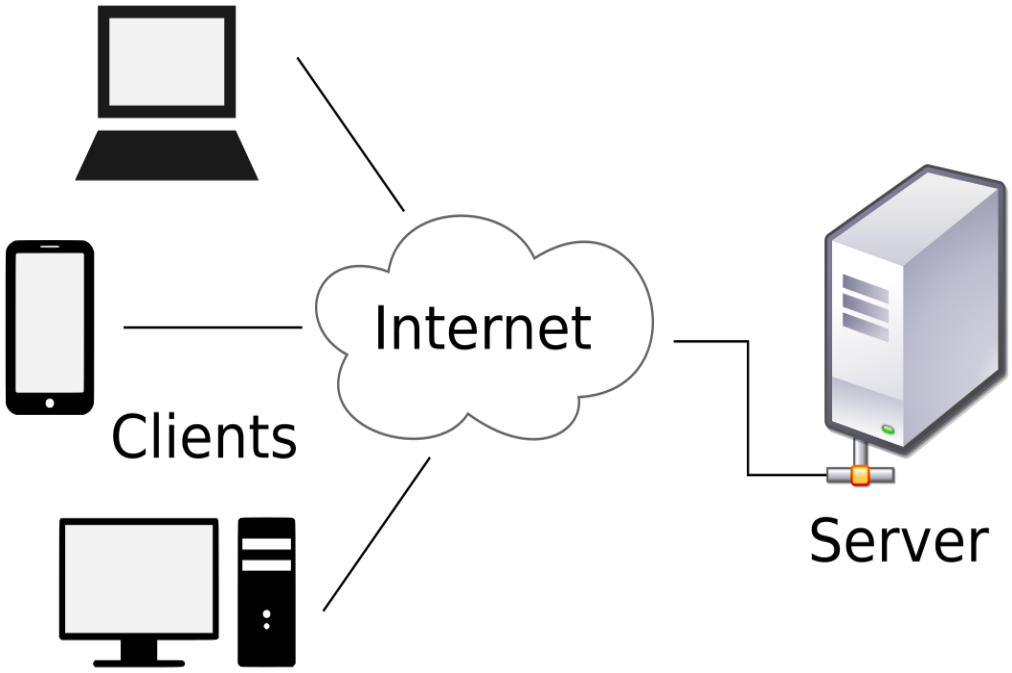


Рис. 4.1. Архітектура додатку

4.1.1. Опис серверної частини додатку

Під серверною частиною додатку треба розуміти базу даних, класи сутностей та сервісні класи.

На рисунку. 4.2 зображено ієрархію класів у створеного додатку.

flow controller

DiplomaApplication точка входу до програми.

DataLoader controller: клас, що запускає у консольному режимі модуль додатку, що відповідає за передініціацію бази даних.

	controller
WebController	клас, що є контролером MVC. Він бере інформацію із бази даних (Model) та направляє її до веб-інтерфейсу (View), де її буде представлено користувачеві.
UserController	клас, що контролює створення та маніпулювання об’єктами юзера, а також їх представлення у клієнта.
PostController	клас, що контролює об’єкти постів.
	model
User	клас-сутність, до якого прив’язується користувач у базі даних.
Post	клас-сутність, що представляє собою пост у системі.
Comment	клас-сутність, що представляє коментар.
Role	перелік ролей, що існують у веб-додатку.
	repository
UserRepository	репозиторій юзерів, що використовується при передініціалізації. Наслідує інтерфейс CrudRepository, що підключає тип доменного об’єкта та його первинний ключ.
PostRepository	репозиторій постів.
CommentRepository	репозиторій коментарів.
	service
UserService	інтерфейс, що впроваджує методи для створення та перегляду користувачів.
PostService	інтерфейс, що надає методи для керування об’єктами постів – їх створення, перегляд, редагування та видалення.
CommentService	інтерфейс, що впроваджує методи для створення та перегляду коментарів.

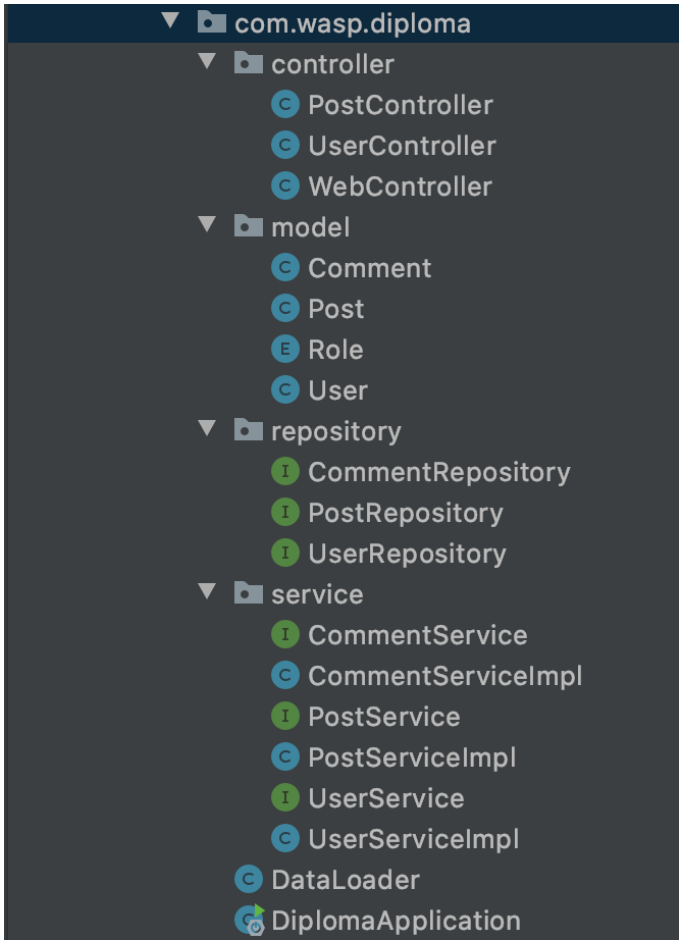


Рис. 4.2. Ієрархія класів у додатку

На рисунку. 4.3 відображено структуру файлів, що виконують такі функції як оголошення залежностей та/або властивостей, що їх буде підтягнуто та завантажено під час компіляції програми.

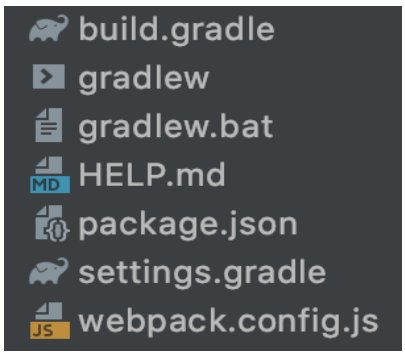


Рис. 4.3 Структура допоміжних файлів властивостей

Серед усіх цих файлів нас на даний момент цікавить лише один:

build.gradle – файл конфігурації Gradle, у якому прописані залежності, які потрібно загрузити для коректної роботи додатку

4.1.2. Опис клієнтської частини додатку

Структура клієнтської частини розробленого веб-додатку зображена на рисунку 4.4.

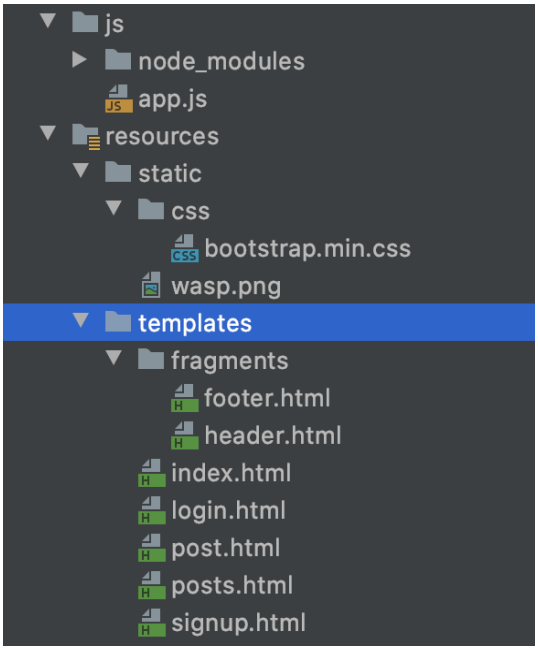


Рис. 4.4. Структура клієнтської частини додатку

Розробку клієнтської частини було проведено з урахуванням модульної системи. Кожен модуль представляє собою одне відображення.

index.html – представляє домашню сторінку

login.html – строрінку логіну

signup.html – сторінку реєстрації

post.html – відкритий пост

posts.html – “стрічку” постів

fragments/header.html – спільна для сторінок шапка

fragments/footer.html – спільна нижня частина сторінки

app.js – файл компонентів додатку, що відповідає за відмальовку компонентів, а також посилання запитів на сервер у реальному режимі зі сторінки.

На рисунку 4.5 зображено структуру файлів, що відповідають за функції ReactJS:

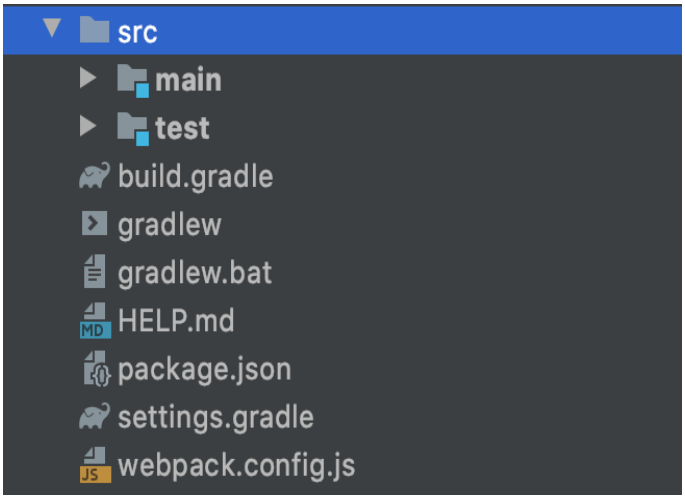


Рис. 4.5. Структура файлів ReactJS

package.json – файл, у якому визначено різноманітні метадані, релевантні у проєкті

webpack.config.js – файл, що визначає app.js точкою входу, создає файл bundle.js, який підставляється у сторінку, під час компіляції.

4.2. Взаємодія користувача із системою

Взаємодія користувача з інтерфейсом починається зі сторінки реєстрації (рис. 4.6, рис. 4.7). На цій сторінці користувач вводить свої дані для реєстрації - електронну пошту, ім’я користувача та пароль.

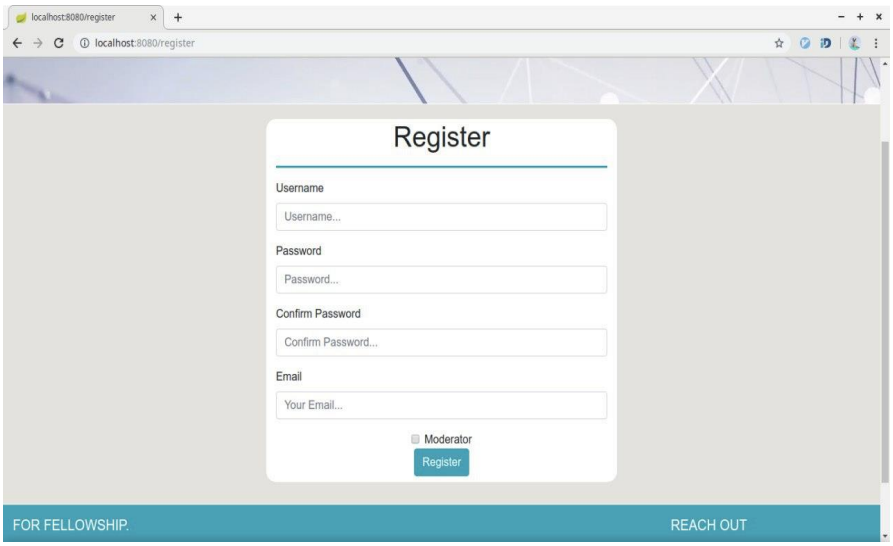


Рис. 4.6. Вигляд сторінки реєстрації до заповнення

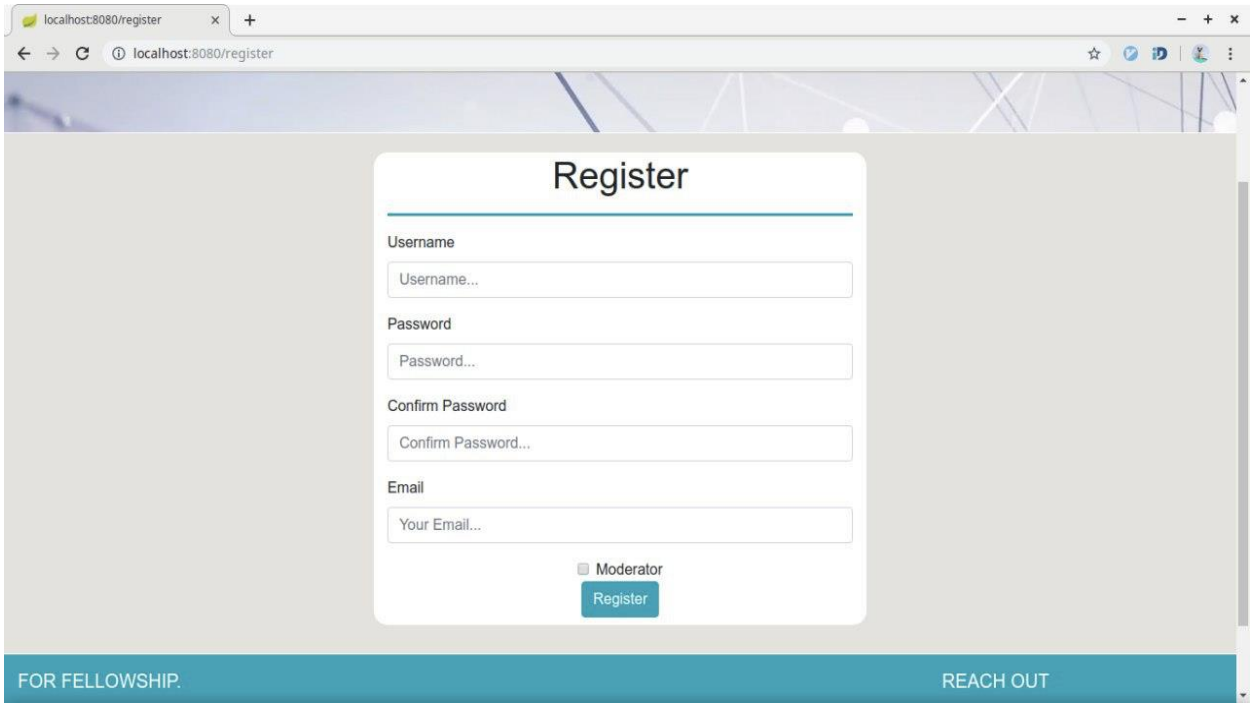


Рис. 4.7. Заповнені поля реєстрації

Якщо користувач вже зареєстрований в системі, для нього наступним кроком буде увійти в неї на сторінці логіну (рис. 4.8 та 4.9).

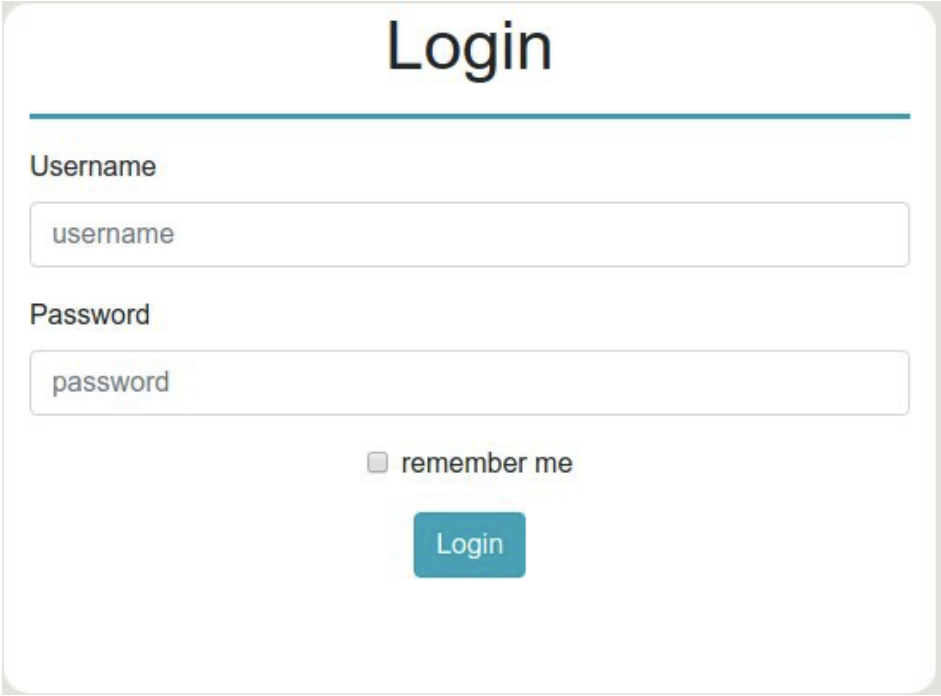


Рис. 4.8. Вигляд сторінки логіну до заповнення

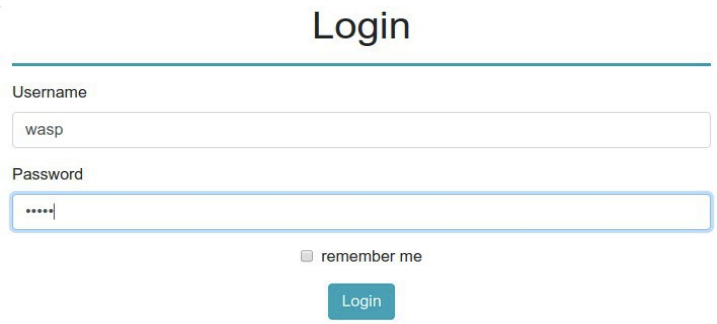


Рис. 4.9. Заповнена сторінка логіну

Пройшовши реєстрацію та/або логін, користувач потрапить на сторінку, на котрій знаходяться останні створені іншими користувачами пости (рис. 4.10). Натиснувши кнопку “Create Post”, користувач потрапляє на сторінку створення нового поста (рис. 4.11).

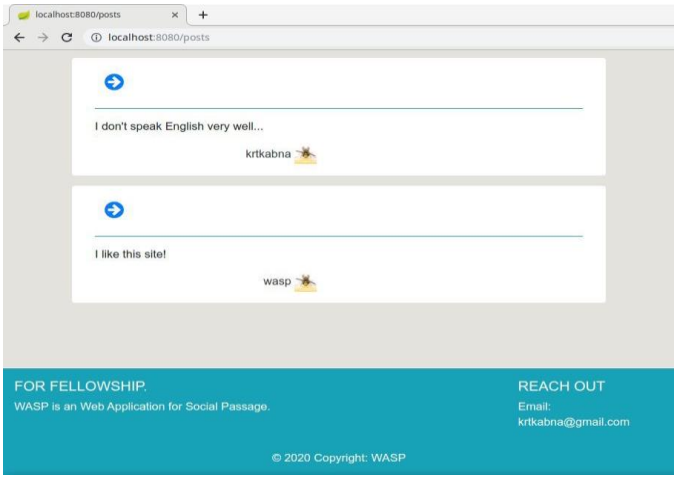


Рис. 4.10. Сторінка постів

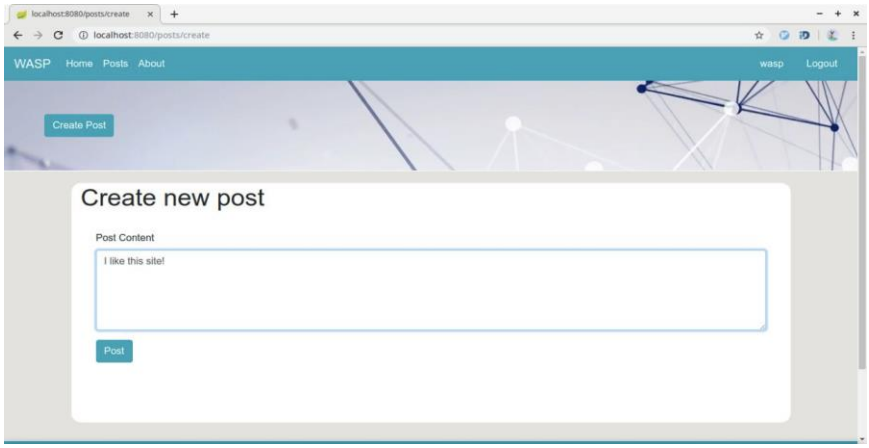


Рис. 4.11. Створення поста

Натискаючи на синю стрілку, користувач переходить на сторінку перегляду поста (рис. 4.12).

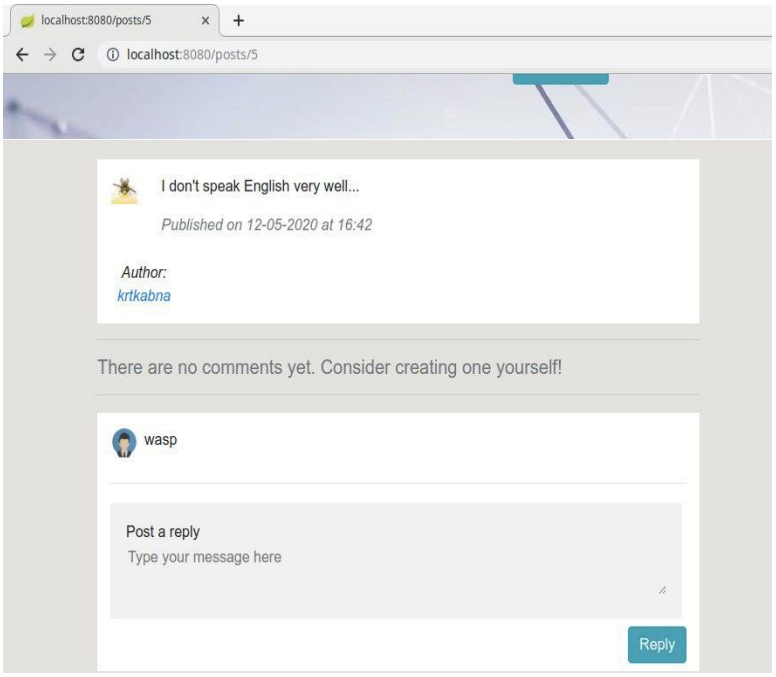


Рис. 4.12. Коментарі відсутні

На попередньому малюнку бачимо, що коментарі для посту поки що відсутні. Створити новий коментар можна, написавши його текст та натиснувши кнопку “Reply”. На рисунку. 4.13 зображено створення коментаря.

Рис. 4.13. Створення коментаря

На рисунку. 4.14 зображено вигляд сторінки посту після додавання до нього коментаря.

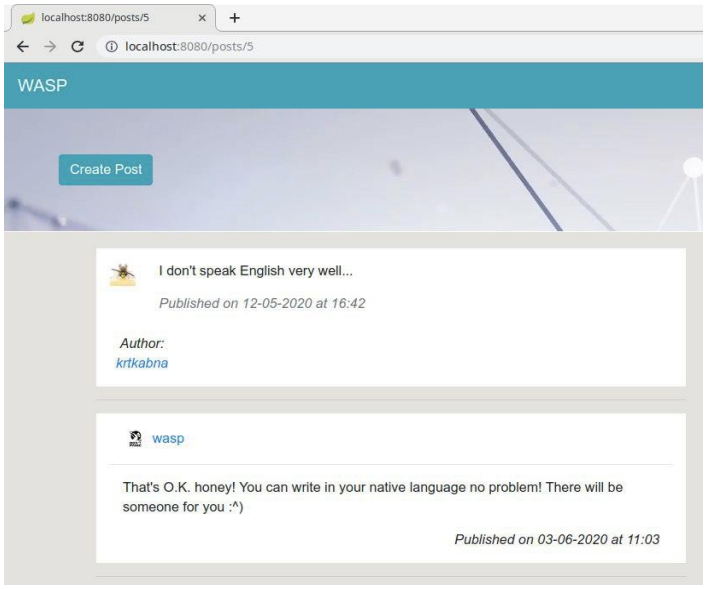


Рис. 4.14. Коментар присутній

Як видно з попередніх малюнків, користувач, що переглядає пост, не є його автором. На рисунку. 4.15 зображений пост очима автора, на якому чітко видно різницю між своєю та чужою публікацією – наявність кнопок “Edit” та “Delete”, що дозволяють відредагувати (рис.4.16) або видалити (рис. 4.17) пост.

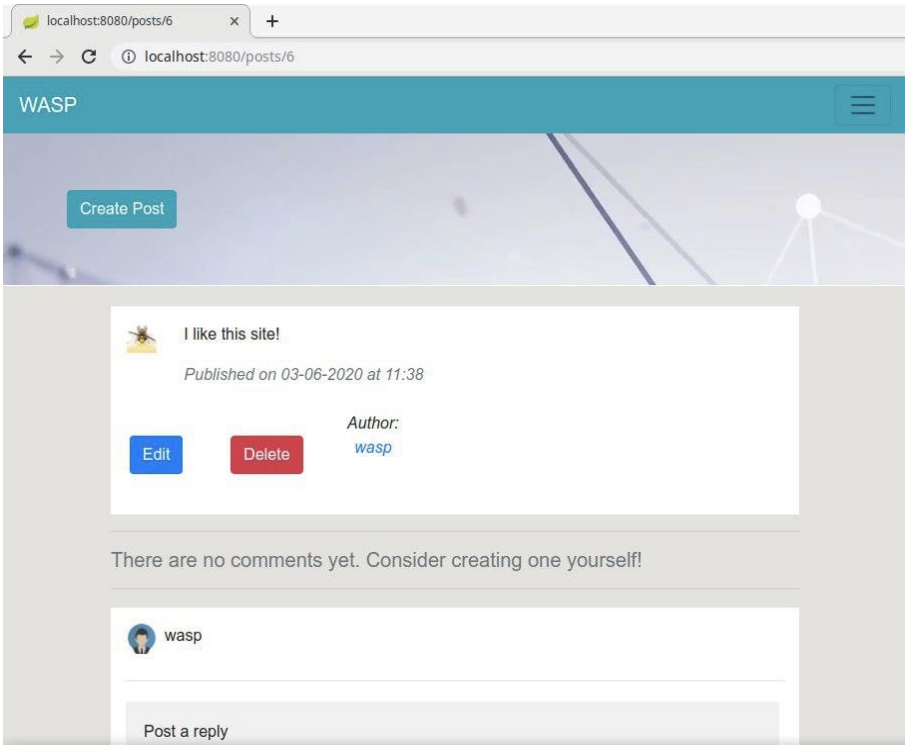


Рис. 4.15. Пост з точки зору його автора

Після натискання кнопки “Edit” користувач бачить спливаюче вікно редагування посту (рис. 4.16).

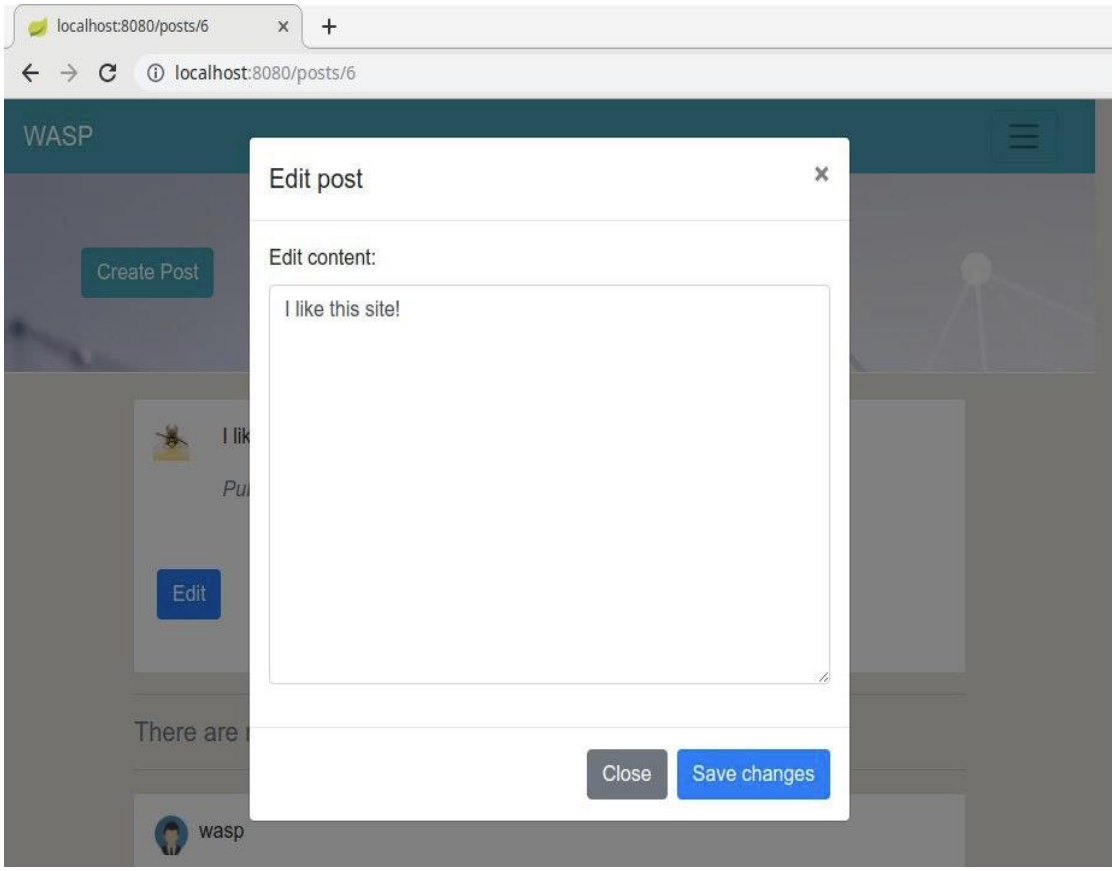


Рис. 4.16. Модальне вікно редагування поста

На рисунку. 4.17 зображено спливаюче вікно підтвердження, що відображається після натискання кнопки “Delete”.

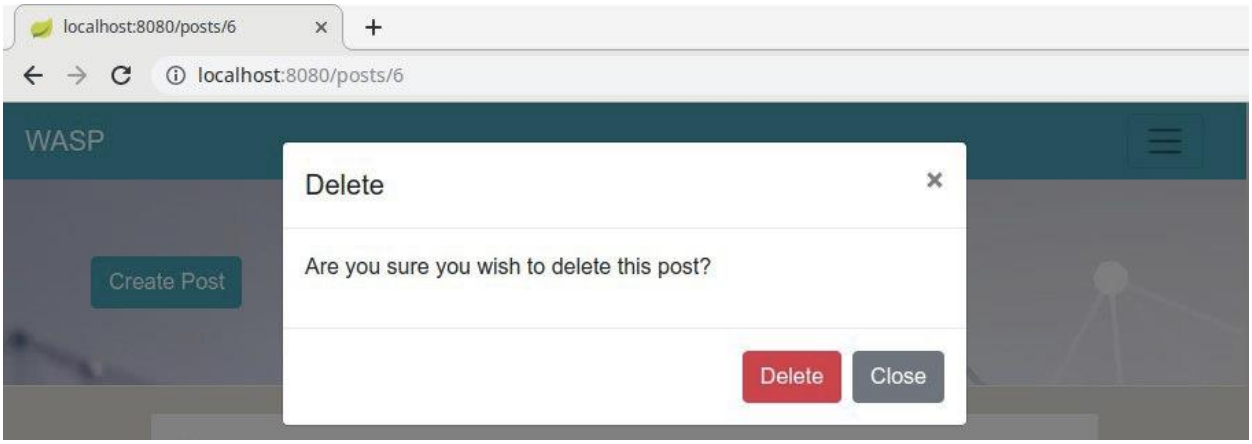


Рис. 4.17. Модальне вікно видалення поста

4.3. Порівняння з аналогами

У порівнянні з існуючими аналогами, описаними в розділі 2, створений додаток є меншим та легшим, а також має наступні переваги:

● **Відсутність у додатку реклами** дозволяє користуватися його функціями, не відволікаючись на спливаючі оголошення, а також робить його швидшим за додатки конкурентів.

● **Відсутність сповіщень** допомагає не відволікатися на додаток під час важливих справ та зустрічей, дозволяє відпочити від додатку та повернутися до нього тоді, коли в цьому дійсно виникне потреба. Додатки конкурентів, постійно відправляючи своїм користувачам купи сповіщень, забирають їхній час та відволікають від важливих речей.

Висновки до розділу 4

У розділі 4 було представлено архітектуру, розроблено програмну реалізацію SNS-додатку за допомогою інструментів мови Java та Spring Framework, а також наведено її переваги перед технологіями-конкурентами.

Програмний додаток дозволяє користувачеві зареєструватися в базі, автентифікуватися, створювати пости та маніпулювати ними, а також залишати коментарі до постів інших користувачів.

Було наведено детальний опис інтерфейсу програми та її логіки, а також розглянуто приклад роботи програми та її використання окремим користувачем. Наведено приклад реєстрації користувача, подальшої автентифікації в системі та використання можливостей розробленої служби соціальних мереж.

Завдяки повній відсутності сповіщень та реклами, розроблений додаток має конкурентні переваги у порівнянні з розглянутими конкурентами.

ВИСНОВКИ

Даний бакалаврський проєкт присвячено створенню веб-додатку, що дозволяє його користувачам спілкуватися один з одним. Пояснювальна записка складається з 4 розділів.

В розділі 1 було проведено аналіз предметної області, розглянуто поняття служби соціальних мереж (SNS) та її характеристику.

Було проведено огляд існуючих рішень та виокремлено найбільш специфічні особливості, а також переваги та недоліки кожного з рішень.

В розділі 2 було розглянуто найбільш поширені технології та фреймворки для написання програмного коду та його підтримки.

Проведено огляд систем управління базами даних, оглянуто декілька популярних реляційних СУБД.

Обрано середою розробки Intellij IDEA Ultimate Edition.

Проведено огляд технологій для написання фронтенду для клієнтської частини програми.

У розділі 3 було визначено конкретні технології, мови та бібліотеки, що будуть використовуватись для програмної реалізації додатку.

Визначено інструменти для розробки як клієнтської, так і серверної частин системи; зокрема під час написання коду програми, а також систему управління базами даних.

З метою ефективного оперування інформацією у базах даних, було обрано REST Repositories та JPA, у якості СУБД обрано H2 Database. У якості серверу для деплойменту запропоновано використання Embedded Apache Tomcat.

Для написання клієнтської частини обрано комбінацію із Thymeleaf, Bootstrap, React та Webpack.

У розділі 4 було представлено архітектуру, розроблено програмну реалізацію SNS-додатку за допомогою інструментів мови Java та Spring Framework, а також наведено її переваги перед технологіями-конкурентами.

Програмний додаток дозволяє користувачеві зареєструватися в базі, автентифікуватися, створювати пости та маніпулювати ними, а також залишати коментарі до постів інших користувачів.

Було наведено детальний опис інтерфейсу програми та її логіки, а також розглянуто приклад роботи програми та її використання окремим користувачем. Наведено приклад реєстрації користувача, подальшої автентифікації в системі та використання можливостей розробленої служби соціальних мереж.

Завдяки повній відсутності сповіщень та реклами, розроблений додаток має конкурентні переваги у порівнянні з розглянутими конкурентами.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. Social networking service [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Social_networking_service.

2. Social Networking Service - SNS Definition [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.investopedia.com/terms/s/social-networking-service-sns.asp.

3. Історія соціальних мереж, пересказана піонером Рунету [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://rb.ru/article/ istoriya-sotssetey-rasskazannaya-pionerom-runeta/7323287.html.

4. Features of Facebook [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/List_of_Facebook_features.

5. Twitter appearance [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Twitter#Appearance_and_features.

6. Pros and Cons of 5 of the Biggest Social Media Platforms [Електронний ресурс] – Режим доступу до ресурсу: https://www.socialmia.com today / content/pros-and-cons-5-biggest-social-media-platforms.

7. Tumblr Blog Management [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Tumblr#Blog_management.

8. A journalist's guide to using Tumblr [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: https://www.poynter.org/reporting-editing/2013/a-journalists-guide-to-using-tumblr/.

9. JCGs (Java Code Geeks). Spring Framework Cookbook: Hot Recipes for Spring Framework / JCGs (Java Code Geeks)., 2017. – 200 с.

10. Система управления базами данных [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%/ D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85.

11. MySQL – Википедия [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/MySQL.

12. What is PostgreSQL? [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresqltutorial.com/what-is-postgresql/.

13. IBM Db2 – Data Management Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/analytics/db2.

14. H2 Database [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/H2_(DBMS).

15. Integrated Development Environment [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Integrated_development _environment.

16. HyperText Markup Language [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/HTML.

17. Thymeleaf [Електронний ресурс] – Режим доступу до ресурсу: https://www.thymeleaf.org/.

18. Introduction to the Angular Docs [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/docs.

19. Getting Started with React [Електронний ресурс] – Режим доступу до ресурсу: https://reactjs.org/docs/getting-started.html.

20. H2 Features [Електронний ресурс] – Режим доступу до ресурсу: https://www.h2database.com/html/features.html.

21. Thymeleaf Features [Електронний ресурс] – Режим доступу до ресурсу: https://web.archive.org/web/20111008055319/http://www.thymeleaf.org /features.html.

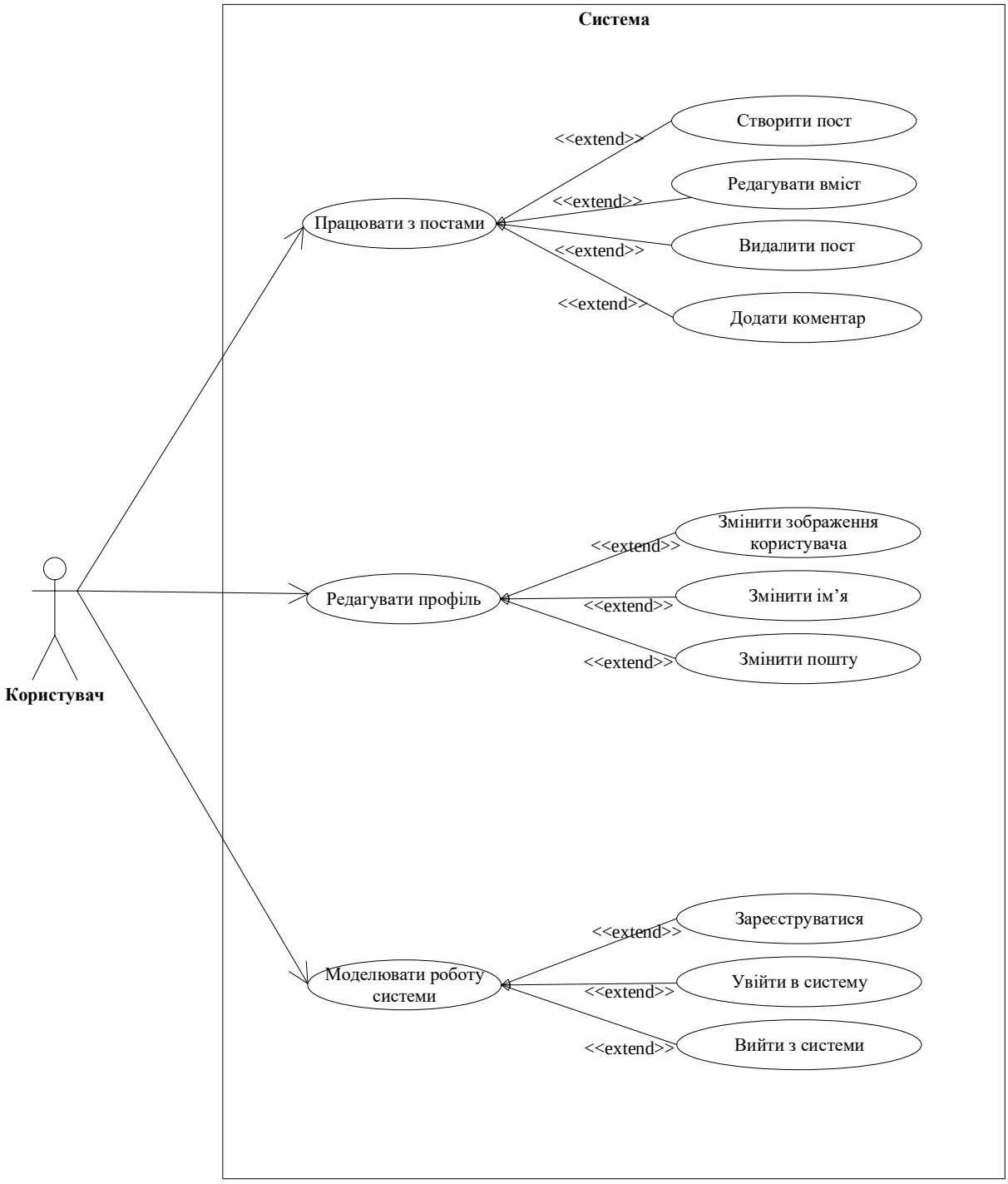
ДОДАТОК 1

ВЕБ-ДОДАТОК ДЛЯ СПІЛКУВАННЯ

СХЕМА РОБОТИ СИСТЕМИ

Аркушів 1

Київ – 2020



					ІАЛЦ.466400.005 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Осипенко А.В.			Веб-додаток для спілкування Технічне завдання	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Регіда П.Г.				КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІП-64
Реценз.						
Н. Контр.		Сімоненко В. П.				
Затвердив						

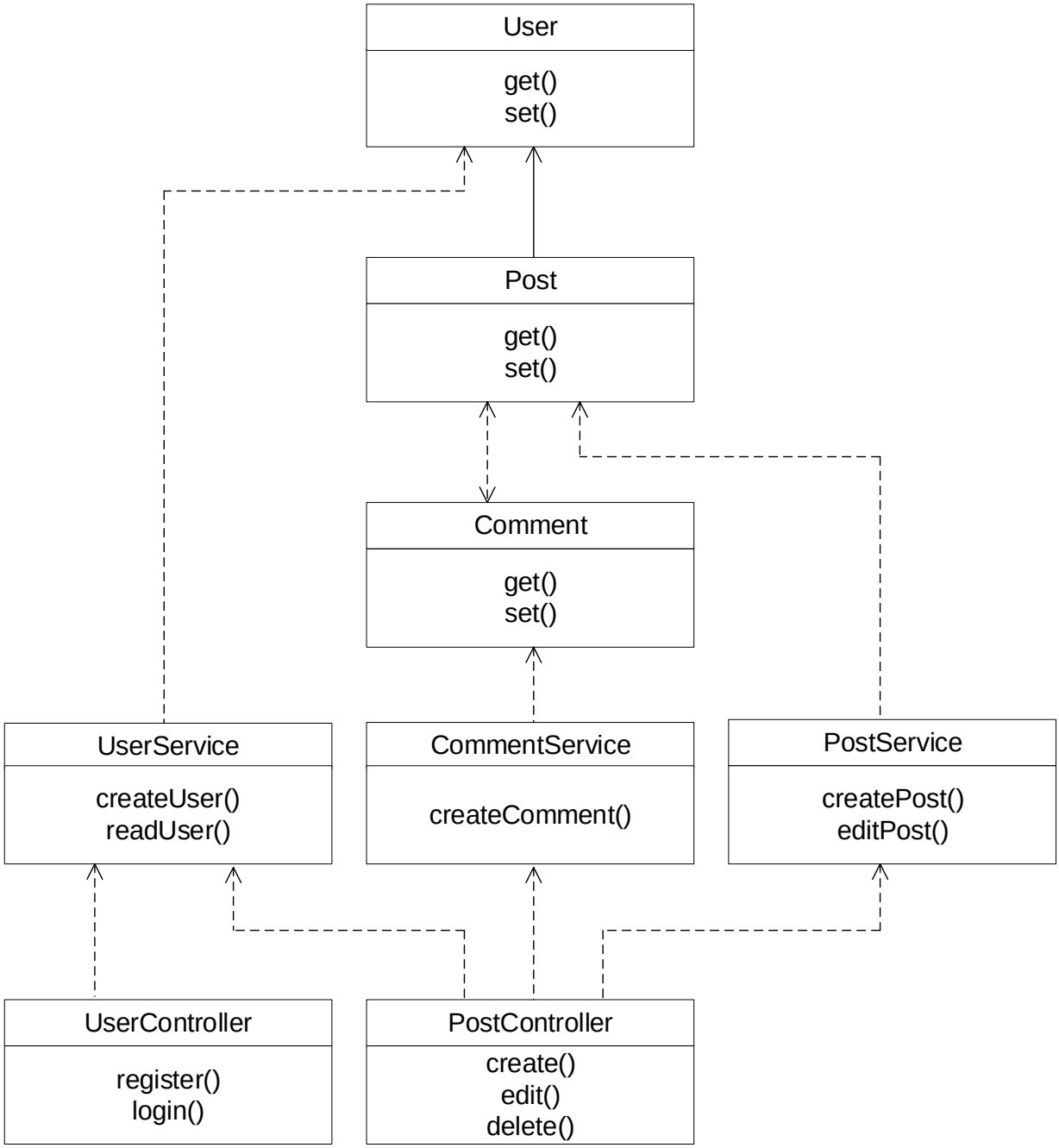
ДОДАТОК 2

ВЕБ-ДОДАТОК ДЛЯ СПІЛКУВАННЯ

СХЕМА ВЗАЄМОДІЇ ПРОГРАМИ

Аркушів 1

Київ – 2020



					ІАЛЦ.466400.006 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Осипенко А.В.			Веб-додаток для спілкування	Літ. / Аркуш / Аркушів
Перевірив		Регіда П.Г.			Технічне завдання	1 / 1
Реценз.						КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІП-64
Н. Контр.		Сімоненко В. П.				
Затвердив						

ДОДАТОК З

ВЕБ-ДОДАТОК ДЛЯ СПІЛКУВАННЯ

СХЕМА ПРОГРАМИ

Аркушів 1

Київ – 2020

					ІАЛЦ.466400.007 ДЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Осипенко А.В.			Веб-додаток для спілкування Технічне завдання	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Регіда П.Г.					
Реценз.						КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІП-64	
Н. Контр.		Сімоненко В. П.					
Затвердив							

ДОДАТОК 4

ВЕБ-ДОДАТОК ДЛЯ СПІЛКУВАННЯ

ЛІСТИНГ ДОДАТКУ

Аркушів 36

Київ – 2020

Клас DiplomaApplication – точка входу у програму

```
package com.wasp.diploma;


import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication
public class DiplomaApplication {


    public static void main(String[] args) {
        SpringApplication.run(DiplomaApplication.class,
args);
    }


}
```

ENTITIES

Клас User

```
package com.wasp.diploma;


import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.util.Objects;



@Entity
public class User {
```

```java
@Id @GeneratedValue(strategy = GenerationType.IDENTITY)
private long id;

private String username;

public User() {

}

public User(String username) {
    this.username = username;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }
    if (o == null || getClass() != o.getClass()) {
        return false;
    }
    User user = (User) o;
```

```
        return id == user.id &&
                Objects.equals(username, user.username);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, username);
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", username='" + username + '\'' +
                '}';
    }
}
```

Клас Post

```
package com.wasp.diploma;


import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
```

```java
import java.time.LocalDateTime;
import java.util.Objects;
import java.util.Set;


@Entity
public class Post {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;

    private String content;

    private LocalDateTime publishDate;

    @ManyToOne
    private User user;

    @OneToMany(
            mappedBy = "post",
            cascade = CascadeType.ALL,
            orphanRemoval = true,
            fetch = FetchType.EAGER
    )
    private Set<Comment> comments;

    public Post() {
    }
```

```
public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public String getContent() {
    return content;
}

public void setContent(String content) {
    this.content = content;
}

public LocalDateTime getPublishDate() {
    return publishDate;
}

public void setPublishDate(LocalDateTime publishDate) {
    this.publishDate = publishDate;
}

public User getUser() {
    return user;
}

public void setUser(User user) {
```

```
        this.user = user;
    }

    public Set<Comment> getComments() {
        return comments;
    }

    public void setComments(Set<Comment> comments) {
        this.comments = comments;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Post)) return false;
        Post post = (Post) o;
        return Objects.equals(getTitle(), post.getTitle()) &&
                Objects.equals(getContent(),
post.getContent()) &&
                Objects.equals(getPublishDate(),
post.getPublishDate()) &&
                Objects.equals(getUser(), post.getUser()) &&
                Objects.equals(getComments(),
post.getComments());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getTitle(), getContent(),
getPublishDate(), getUser(), getComments());
    }
```

```
    @Override
    public String toString() {
        return "Post{" +
                "id=" + id +
                ", title='" + title + '\'' +
                ", content='" + content + '\'' +
                ", publishDate=" + publishDate +
                ", user=" + user +
                ", comments=" + comments +
                '}';
    }

}
```

Клас Comment

```
package com.wasp.diploma.model;


import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import java.time.LocalDateTime;
import java.util.Objects;


@Entity
public class Comment {
```

```java
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

private String content;

private LocalDateTime publishDate;

@ManyToOne(fetch = FetchType.EAGER)
private Post post;

@ManyToOne(fetch = FetchType.EAGER)
private User user;

public Comment() {
}

public Comment(String content) {
    this.content = content;
}

 public String getContent() {
         return content;
 }

 public void setContent(String content) {
         this.content = content;
 }
```

```
        public LocalDateTime getPublishDate() {
                return publishDate;
        }

        public void setPublishDate(LocalDateTime
publishDate) {
                this.publishDate = publishDate;
        }

        public Post getPost() {
                return post;
        }

        public void setPost(Post post) {
                this.post = post;
        }

        public User getUser() {
                return user;
        }

        public void setUser(User user) {
                this.user = user;
        }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Comment)) return false;
        Comment comment = (Comment) o;
```

```
        return Objects.equals(getContent(),
comment.getContent()) &&
                Objects.equals(getPublishDate(),
comment.getPublishDate()) &&
                Objects.equals(getPost(), comment.getPost())
&&
                Objects.equals(getUser(), comment.getUser());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getContent(), getPublishDate(),
getPost(), getUser());
    }

    @Override
    public String toString() {
        return "Comment{" +
                "id=" + id +
                ", content='" + content + '\'' +
                ", publishDate=" + publishDate +
                ", post=" + post +
                ", user=" + user +
                '}';
    }
}
```

Клас Role

```
package com.wasp.diploma.model;

import org.springframework.security.core.GrantedAuthority;
```

```
public enum Role implements GrantedAuthority {
    MODERATOR, AUTHENTICATED, UNAUTHENTICATED;

    public String getAuthority() {
        return name();
    }

}
```

REPOSITORIES

Клас UserRepository

```
package com.wasp.diploma.repository;

import com.wasp.diploma.model.User;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepository extends CrudRepository<User,
Long> {
}
```

Клас PostRepository

```
package com.wasp.diploma.repository;

import com.wasp.diploma.model.Post;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

@Repository
```

```
public interface PostRepository extends CrudRepository<Post,
Long> {

}
```

Клас CommentRepository

```
package com.wasp.diploma.repository;


import com.wasp.diploma.model.Comment;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;


@Repository

public interface CommentRepository extends
CrudRepository<Comment, Long> {

}
```

Клас UserService

```
package com.wasp.diploma.service;


import com.wasp.diploma.model.User;


public interface UserService {

    User createUser(User user);


    User readUser(User user);


}
```

SERVICES

Клас UserServiceImpl

```
package com.wasp.diploma.service;


import com.wasp.diploma.model.User;
import com.wasp.diploma.model.Role;



public class UserServiceImpl implements UserService {
     private final UserRepository userRepository;
     private final RoleService roleService;


     @Autowired
public UserServiceImpl(UserRepository userRepository,
RoleService roleService) {
    this.userRepository = userRepository;
    this.roleService = roleService;
}


    @Override
    public void createUser(User user) {
     RoleServiceModel roleServiceModel =
this.roleService.findByAuthority(userServiceModel.isModerator
() ? "MODERATOR" : "USER");
Role role  = this.modelMapper.map(roleServiceModel,
Role.class);


user.addRole(role);


       this.userRepository.save(user);
    }


    @Override
    public void readUser() {
```

```
        this.postRepository.delete(post);    }
}
```

Клас PostService

```
package com.wasp.diploma.service;

import com.wasp.diploma.model.Post;

public interface PostService {
    void createPost();

    void editPost(Post post, String changes);

    void deletePost(Post post);

}
```

Клас PostServiceImpl

```
package com.wasp.diploma.service;

import com.wasp.diploma.model.Post;
import com.wasp.diploma.repository.PostRepository;
import
org.springframework.beans.factory.annotation.Autowired;

public class PostServiceImpl implements PostService {
    private final PostRepository postRepository;

    public PostServiceImpl() {
```

```
    }

    @Autowired
    public PostServiceImpl(PostRepository postRepository) {
        this.postRepository = postRepository;
    }

    @Override
    public Post editPost(Post post, String changes) {
        post.setContent(changes);
        this.postRepository.save(post);
    }

    @Override
    public void deletePost(Post post) {
        this.postRepository.delete(post);
    }
}
```

Клас CommentService

```
package com.wasp.diploma.service;

import com.wasp.diploma.model.Comment;

public interface CommentService {
    void createComment();

}
```

Клас CommentServiceImpl

```
package com.wasp.diploma.service;


import com.wasp.diploma.model.Comment;

import com.wasp.diploma.repository.CommentRepository;

import
org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;


import java.time.LocalDateTime;


@Service

public class CommentServiceImpl implements CommentService {


    private final CommentRepository commentRepository;


    @Autowired

    public CommentServiceImpl(CommentRepository
commentRepository) {

        this.commentRepository = commentRepository;

   }


    @Override

    public void create(Comment comment) {


commentEntity.setPublishDate(LocalDateTime.now());

        this.commentRepository.save(commentEntity);

    }

}
```

CONTROLLERS

Клас WebController

```
package com.wasp.diploma.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class WebController {

    @RequestMapping(value = "/")
    public String index() {
        return "index";
    }

    @GetMapping(value = "/login")
    public String getLogin() {
        return "login";
    }

    @GetMapping(value = "/signup")
    public String getSignUp() {
        return "signup";
    }

    @GetMapping(value = "/about")
    public String getAbout() {
        return "login";
    }
```

```
}
```

Клас UserController

```
package com.wasp.diploma.controller;


import com.wasp.diploma.service.UserService;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.ModelAndView;


import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;


public class UserController {
    private final UserService userService;


    private final ModelMapper modelMapper;


    @Autowired
    public UserController(UserService userService,
ModelMapper modelMapper) {
        this.userService = userService;
        this.modelMapper = modelMapper;
    }
```

```
    @GetMapping("/register")
    public ModelAndView register(@ModelAttribute
UserRegisterBindingModel userRegisterBindingModel) {
        return super.view("views/users/register",
"Register");
    }

    @PostMapping("/register")
    public ModelAndView registerConfirm(@Valid
@ModelAttribute UserRegisterBindingModel
userRegisterBindingModel,
                                        BindingResult
bindingResult,
                                        HttpServletRequest
request) {

        if (bindingResult.hasErrors()) {
            return super.view("views/users/register",
"Register");
        }

        UserServiceModel userServiceModel =
this.modelMapper.map(userRegisterBindingModel,
UserServiceModel.class);
        this.userService.createUser(userServiceModel);
        return super.redirect("/login");
    }

    @GetMapping("/users/{username}")
    public ModelAndView userProfile(@PathVariable String
username) {
        UserServiceModel userServiceModel =
this.userService.findByUsername(username);
```

```
        return super.view("/views/users/profile",
userViewModel);
    }

    @GetMapping("/login")
    public ModelAndView login(String error, ModelAndView mav)
{
        mav.addObject("viewName", "/views/users/login");
        mav.setViewName("layout");
        if (error != null) {
            mav.addObject("error", "Wrong username or
password");
        }
        return mav;
    }
}
```

Клас PostController

```
package com.wasp.diploma.controller;



import com.wasp.diploma.model.Post;
import com.wasp.diploma.service.CommentService;
import com.wasp.diploma.service.PostService;
import com.wasp.diploma.service.UserService;
import org.modelmapper.ModelMapper;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageImpl;
import org.springframework.data.domain.Pageable;
```

```java
import org.springframework.data.web.PageableDefault;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;

import javax.validation.Valid;
import java.util.ArrayList;

@Controller
@RequestMapping("/posts")
public class PostController extends BaseController {

    private final PostService postService;

    private final UserService userService;

    private final CommentService commentService;

    private final ModelMapper modelMapper;

    @Autowired
    public PostController(PostService postService,
UserService userService, CommentService createCommentDto,
ModelMapper modelMapper) {
        this.postService = postService;
        this.userService = userService;
        this.commentService = createCommentDto;
        this.modelMapper = modelMapper;
    }
```

```
    @GetMapping("")
    public ModelAndView allPosts(@PageableDefault(size = 10)
Pageable pageable, @RequestParam(value = "search", required =
false) String searchGetParameter) {
        Page<PostServiceModel> postServiceModels;
        if (searchGetParameter != null) {
            postServiceModels =
this.postService.findAllByName(searchGetParameter, pageable);
        } else {
            postServiceModels =
this.postService.findAll(pageable);
        }

        List<PostViewModel> postViewModelList = new
ArrayList<>();
        postServiceModels.forEach(postServiceModel -> {
            PostViewModel postViewModel =
this.modelMapper.map(postServiceModel, PostViewModel.class);
            postViewModelList.add(postViewModel);
        });

        Page<PostViewModel> postViewModelPages = new
PageImpl<PostViewModel>(postViewModelList, pageable,
postServiceModels .getTotalElements());
        return super.view("views/posts/all",
postViewModelPages);
    }

    @GetMapping("/create")
    public ModelAndView createPost(@ModelAttribute
CreatePostBindingModel createPostBindingModel) {
        List<PostViewModel> postViewModels = new
ArrayList<>();
```

```
        this.postService.findAll().forEach(postServiceModel
-> {
            PostViewModel postViewModel =
                    this.modelMapper.map(postServiceModel,
Post.class);
            postViewModels.add(postViewModel);
        });
        return super.view("views/posts/create",
postViewModels);
    }


    @PostMapping("/create")
    public ModelAndView storePost(@Valid @ModelAttribute
CreatePostBindingModel createPostBindingModel, BindingResult
bindingResult, Authentication authentication) {


        PostServiceModel postServiceModel =
this.modelMapper.map(createPostBindingModel,
PostServiceModel.class);


        UserServiceModel userServiceModel =
this.userService.findByUsername(authentication.getName());
        postServiceModel.setUser(userServiceModel);
        this.postService.create(postServiceModel);
        return super.redirect("/posts");
    }


    @PostMapping("/{id}")
    public ModelAndView storeAnswer(@Valid @ModelAttribute
CommentBindingModel commentBindingModel, BindingResult
bindingResult, Authentication authentication,
                                    @PathVariable(value =
"id", required = true) Long postId) {
```

```
        CommentServiceModel commentServiceModel = new
CommentServiceModel();


commentServiceModel.setContent(commentBindingModel.getComment
Content());
        PostServiceModel postServiceModel =
this.postService.findById(postId);
        commentServiceModel.setPost(postServiceModel);
        UserServiceModel userServiceModel =
this.userService.findByUsername(authentication.getName());
        commentServiceModel.setUser(userServiceModel);
        this.commentService.create(commentServiceModel);
        return super.redirect("/posts/" + postId);
    }


    @PostMapping("/{id}/edit")
    public ModelAndView editConfirm(@Valid @ModelAttribute
EditPostBindingModel editPostBindingModel, BindingResult
bindingResult, @PathVariable Long id) {
        if (bindingResult.hasErrors()) {
            return super.redirect("/posts/" + id);
        }
        PostServiceModel postServiceModel =
this.modelMapper.map(editPostBindingModel,
PostServiceModel.class);
        this.postService.edit(postServiceModel);
        return redirect("/posts/" + id);
    }


    @PostMapping("/{id}/delete")
    public ModelAndView deleteConfirm(@PathVariable Long id)
{
        this.postService.deleteById(id);
        return redirect("/posts");
```

```
    }

}
```

DATALOADER

Клас DataLoader

```
package com.wasp.diploma;

import com.wasp.diploma.model.User;
import com.wasp.diploma.repository.UserRepository;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class DataLoader implements CommandLineRunner {

    private final UserRepository repository;

    @Autowired public DataLoader(UserRepository repository) {
        this.repository = repository;
    }

    @Override
    public void run(String... args) throws Exception {
        this.repository.save(new User("VespulaVulgaris"));
    }
}
```

WEB INTERFACE

Клас app.js

```
const React = require('react');
const ReactDOM = require('react-dom');

class App extends React.Component {

  constructor(props) {
    super(props);
    this.state = {users: []};
  }

  componentDidMount() {
    fetch('http://localhost:8080/wasp/users').then(response
=> {
      this.setState({users:
response.entity._embedded.users});
    });
  }

  render() {
    return (
        <UserList users={this.state.users}/>
    )
  }
}

class UserList extends React.Component {
  render() {
    const users = this.props.users.map(user =>
        <User key={user._links.self.href} user={user}/>
    );
```

```
    return (
        <table>
          <tbody>
          <tr>
            <th>User Name</th>
          </tr>
          {users}
          </tbody>
        </table>
    )
  }
}

class User extends React.Component{
  render() {
    return (
        <tr>
          <td>{this.props.user.username}</td>
        </tr>
    )
  }
}

ReactDOM.render(
    <App />,
    document.getElementById('react')
)
```

Файл application.properties

```
spring.data.rest.base-path=/wasp
```

FRAGMENTS

Фрагмент header.html

```
<div class="header-nav p-5" xmlns:th="http://www.w3.org/1999/
xhtml">
    <div class="container-fluid row">
        <form class="form-inline offset-sm-2 col-sm-6"
method="get" action="/posts">
            <input name="search" class="form-control mr-sm-2"
type="search" th:placeholder="#{header.search}" aria-
label="Search">
            <button class="btn btn-outline-info my-2 my-sm-0"
type="submit" th:text="#{header.search.button}"></button>
        </form>
        <a class="nav-link text-white col-sm-4" th:href="@{/
posts/create}">
            <button class="btn btn-info header-button"
type="button" th:text="#{header.create.post}">
            </button>
        </a>
    </div>
</div>
```

Фрагмент footer.html

```
<footer class="page-footer font-small text-white border-top
border-white bg-info" xmlns:th="http://www.w3.org/1999/
xhtml">
    <div class="container-fluid text-center text-md-left
pt-3">
        <div class="row">
            <div class="col-md-6 mt-md-0 mt-3">
                <h5 class="text-uppercase">For fellowship.</
h5>
```

```
                <p th:text="#{footer.description}"></p>
            </div>
            <hr class="clearfix w-100 d-md-none pb-3">
            <div class="col-md-3 mb-md-0 mb-3"></div>
            <div class="col-md-3 mb-md-0 mb-3">
                <h5 class="text-uppercase">Reach out</h5>
                <ul class="list-unstyled">
                    <li>
                        <p>Email: krtkabna@gmail.com</p>
                    </li>
                </ul>
            </div>
        </div>
    </div>
    <div class="footer-copyright text-center py-3">© 2020
Copyright:
        <a class="text-white" th:href="@{/}">WASP</a>
    </div>
</footer>
```

PAGES

Сторінка index.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <link href="css/bootstrap.min.css" rel="stylesheet"/>
  <title>Home</title>
</head>
<body>
```

```
  <!--
============================================================
============== -->

  <!-- This content is only used for static prototyping
purposes (natural templates)-->

  <!-- and is therefore entirely optional, as this markup
fragment will be included -->

  <!-- from "fragments/header.html" at runtime.
-->

  <!--
============================================================
============== -->
  <div th:insert="fragments/general.html :: header"/>
  <h1>Hi beech!</h1>
  <div id="react"></div>
  <script src="bundle.js"></script>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    Praesent scelerisque neque neque, ac elementum quam
dignissim interdum.
    Phasellus et placerat elit. Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
    Praesent scelerisque neque neque, ac elementum quam
dignissim interdum.
    Phasellus et placerat elit.
  </p>
  <footer th:insert="fragments/general.html :: footer"></
footer>
</body>
</html>
```

Сторінка login.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
```

```
<head>
  <meta charset="UTF-8">
  <link href="css/bootstrap.min.css" rel="stylesheet"/>
  <title>Log In</title>
</head>
<body>
<th:block th:replace="fragments/general.html :: header"/>
<div class="container">
  <div class="row justify-content-center">
    <form>
      <div class="form-group">
        <label for="exampleInputUsername1">Username</label>
        <input aria-describedby="usernameHelp" class="form-
control" id="exampleInputUsername1"
               placeholder="username" type="text">
      </div>
      <div class="form-group">
        <label for="exampleInputPassword1">Password</label>
        <input class="form-control"
id="exampleInputPassword1" placeholder="password"
               type="password">
      </div>
      <button class="btn btn-primary" type="submit">Submit</
button>
    </form>
  </div>
  <div class="row"></div>
  <p id="login-greeting">This content will be replaced by
Spring Security.</p>
  <th:block th:replace="fragments/general.html :: footer"/>
</div>
</body>
```

```
</html>
```

Сторінка signup.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <link href="css/bootstrap.min.css" rel="stylesheet"/>
  <title>Sign Up</title>
</head>
<body>
<th:block th:replace="fragments/general.html :: header"/>
<div class="container">
  <div class="row justify-content-center">
    <form>
      <div class="form-group">
        <label for="exampleInputEmail1">Email address</label>
        <input aria-describedby="emailHelp" class="form-
control" id="exampleInputEmail1"
               placeholder="Enter email"
               type="text">
        <small class="form-text text-muted"
id="emailHelp">We'll never share your email with anyone
          else.</small>
      </div>
      <div class="form-group">
        <label for="exampleInputUsername1">Username</label>
        <input aria-describedby="usernameHelp" class="form-
control" id="exampleInputUsername1"
               placeholder="username" type="text">
      </div>
```

```
        <div class="form-group">
          <label for="exampleInputPassword1">Password</label>
          <input class="form-control"
id="exampleInputPassword1" placeholder="Password"
                 type="password">
        </div>
        <button class="btn btn-primary" type="submit">Submit</
button>
      </form>
    </div>
</div>
<th:block th:replace="fragments/general.html :: footer"/>
</body>
</html>
```

Сторінка posts.html

```
<div class="container" xmlns:th="http://www.thymeleaf.org">
    <div class="row">
        <div class="col-lg-8">

            <!-- RENDER POSTS -->
            <th:block th:each="post : ${viewModel}">
                <th:block th:include="~{fragments/posts/all/
single-post}"/>
            </th:block>

            <!-- PAGINATION -->
            <div th:if="${viewModel.getTotalElements()} > 0">
                <th:block th:include="~{fragments/posts/all/
pagination}"/>
            </div>
```

```
            <!-- MESSAGE FOR MISSING POSTS-->
            <th:block th:include="~{fragments/posts/all/
message}"/>

        </div>
        <th:block th:include="~{fragments/sidebar}"/>
    </div>
</div>
```

Сторінка post.html

```
<div class="bg-white p-3">
    <div class="row">
        <div class="avatar col-sm-1 col-md-1">
            <img src="/img/wasp1.jpg" height="512"
width="512"/></div>
        <div class="col-xs-11 col-sm-11 col-md-11 text-">
            <div class="row">
                <h4 class="col-12 post-title"
th:text="*{title}"></h4>
                <i class="text-left col-12 text-secondary
mb-4"
                   th:text="|Published on
*{#temporals.format(publishDate, 'dd-MM-yyyy')} at
*{#temporals.format(publishDate, 'HH:mm')}|"></i>
            </div>
            <p class="text-wrap" th:text="*{content}"></p>
        </div>
    </div>
        <div class="col-md-6 text-center row">
            <th:block th:if="${#authentication.name} ==
*{user.username}">
                <div class="col-md-8 d-flex justify-content-
center">
```

```
                    <button data-toggle="modal" data-
target="#editModal"
                                class="btn m-4 text-white bg-
primary text-center">
                        Edit
                    </button>
                    <button class="btn m-4 btn-danger text-
center"
                                data-toggle="modal" data-
target="#deleteModal">
                        Delete
                    </button>
                </div>
            </th:block>
            <th:block th:unless="${#authentication.name} ==
*{user.username}">
                <div
sec:authorize="hasAuthority('MODERATOR')">
                    <div class="col-md-8 d-flex justify-
content-center">
                        <button class="btn m-1 btn-danger
text-center"
                                    data-toggle="modal" data-
target="#deleteModal">
                            Delete
                        </button>
                    </div>
                </div>
            </th:block>
            <th:block th:unless="${#authentication.name} ==
*{user.username}">
                <div class="offset-md-8"></div>
            </th:block>
            <div class="text-center col-md-4 text-center">
```

```
                <i>
                    Author: <a th:href="@{/users/{username}
(username=*{user.username})}"
                               th:text="*{user.username}"></
a>
                </i>
            </div>
        </div>
    </div>
</div>
```